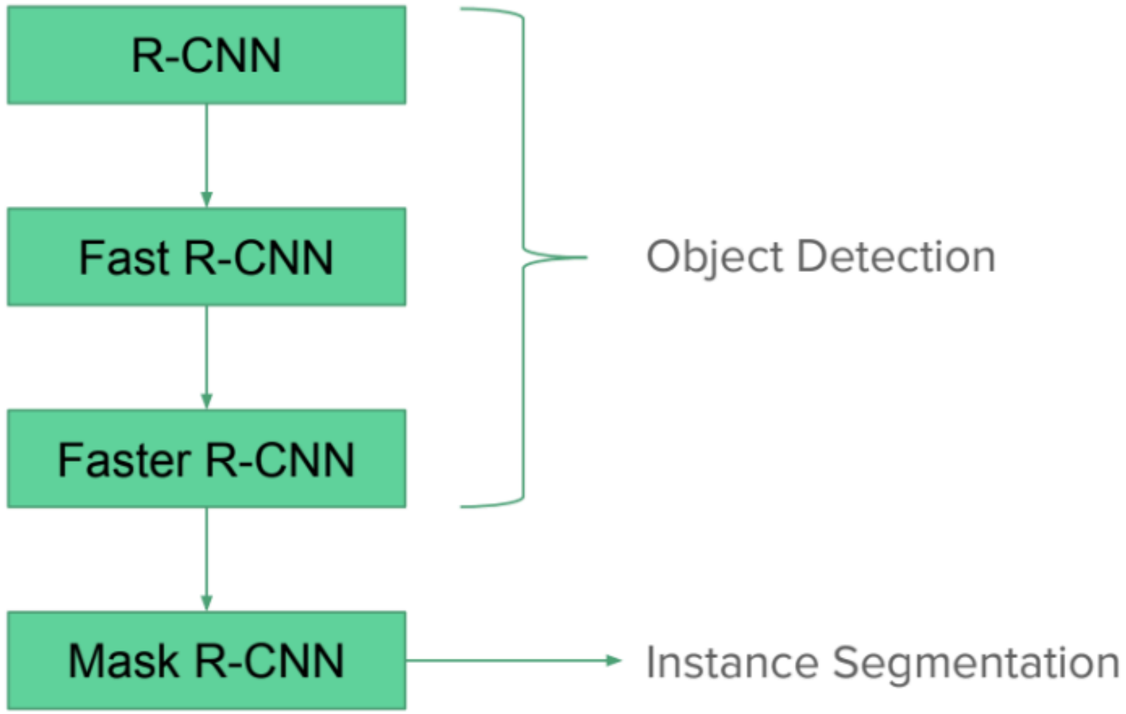
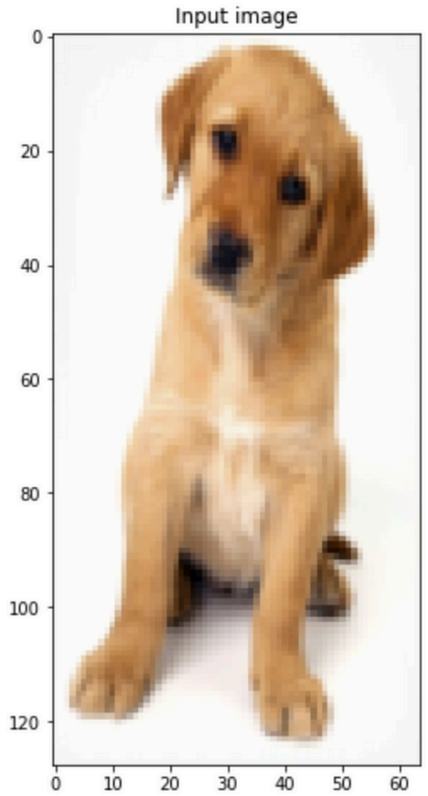
Evolution of R-CNN

The R-CNN Family



History (before R-CNN)

- The previous decade of progress on various visual recognition tasks had been considerably on using SIFT (Scale Invariant Feature Transform) and HOG (Histogram of Oriented Gradients).
- Both HOG and SIFT are blockwise orientation histograms and rely on gradients to identify local features in an image. Gradients represent changes in intensity between pixels and often correspond to edges and corners, which are informative for distinguishing objects.



HOG focuses on edges, as they are informative for distinguishing objects.

Histograms of Oriented Gradients for Human Detection Navneet Dalal and Bill Triggs

- Fukushima's neocognitron was a biologically inspired hierarchical shift-invariant model for pattern recognition.
- It was an early attempt at a hierarchical, multi-stage process for computing features that are even more informative for visual recognition.

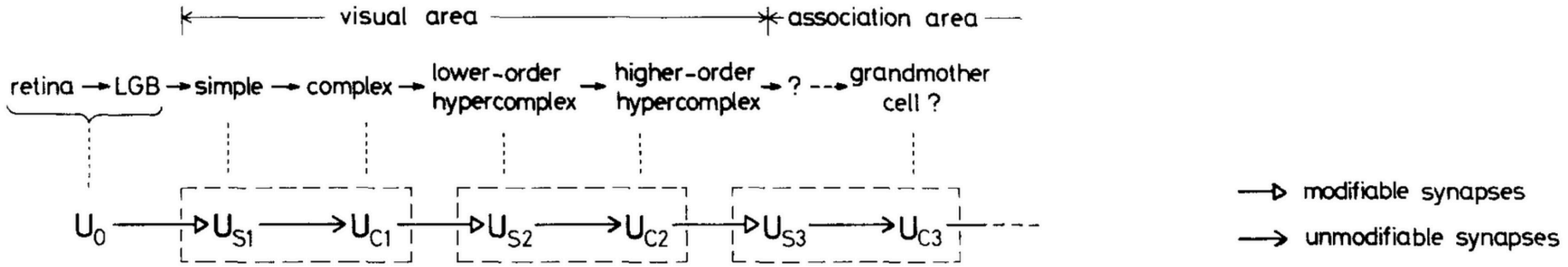
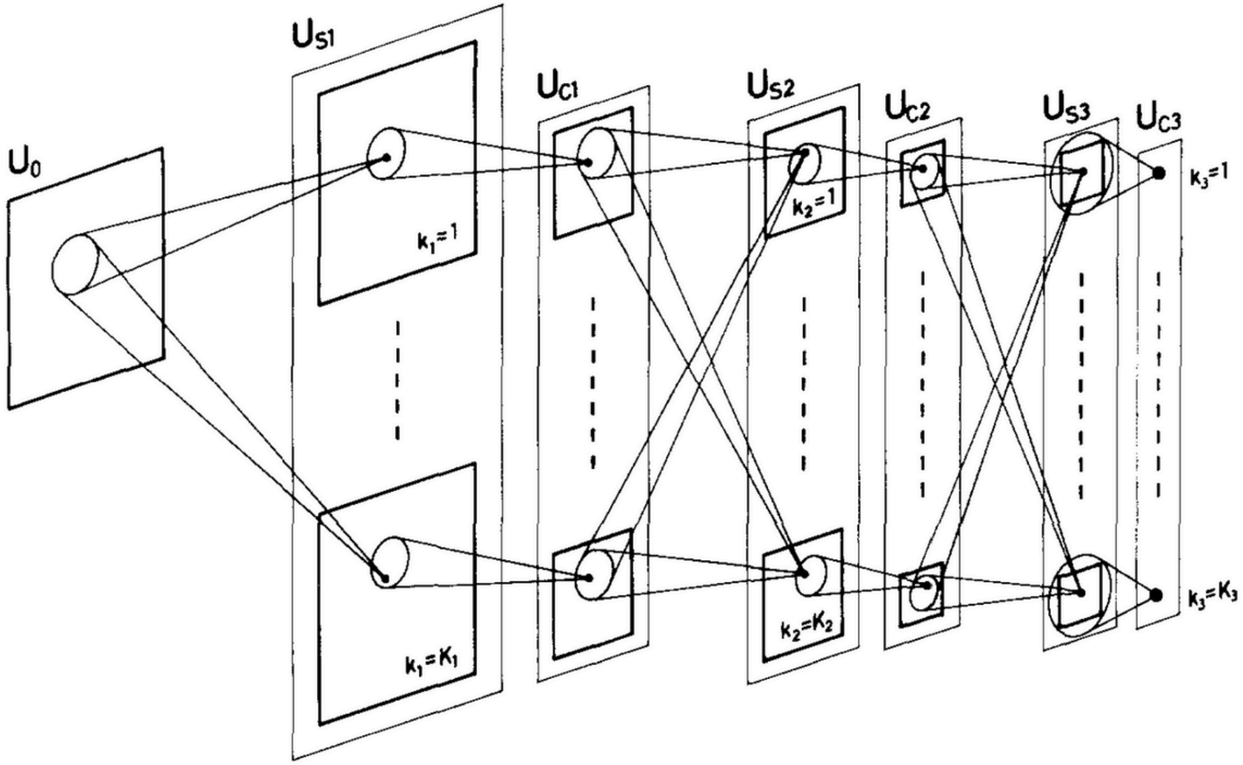


Fig. 1. Correspondence between the hierarchy model by Hubel and Wiesel, and the neural network of the neocognitron



Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position Kunihiko Fukushima

Fig. 2. Schematic diagram illustrating the interconnections between layers in the neocognitron

- Rumelhart et al.'s demonstration of backpropagation for MLPs (Multi-Layer Perceptrons, a precursor to CNNs) and LeCun's successful application to CNNs provided strong evidence that Stochastic Gradient Descent (SGD) with backpropagation could effectively train deeper architectures like CNNs.
- CNNs saw heavy use in the 1990s but then fell out of fashion with the rise of Support Vector Machines (SVMs).
- Interest in CNNs rekindled when Krizhevsky et al. showed substantially higher image classification accuracy on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) by training a large CNN on 1.2 million labeled images, together with a few twists on LeCun's CNN (eg. ReLU and Dropout Regularization).
- The significance of this result was vigorously debated in the ILSVRC workshop 2012. The main topic of discussion of that workshop can be summarized as: **To what extent do the CNN classification results on ImageNet generalize to object detection results on the PASCAL VOC Challenge?**

Rich feature hierarchies for accurate object detection and semantic segmentation
Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik

- R-CNN answers this question by bridging the gap between image classification and object detection.
- R-CNN shows that a CNN can lead to dramatically higher object detection performance on PASCAL VOC as compared to systems based on simpler HOG-like features.

To achieve this result, they focus on 2 main problems :-

1. **Localizing objects with a deep network.**
2. **Training a high capacity model with only a small quantity of annotated detection data.**

Let’s look at the first problem : Localizing objects with a deep network.

- One approach is to frame localization as a regression problem. However, work from Szegedy et al. indicates that this strategy does not fare well in practise (mAP of 30.5% on VOC 2007).

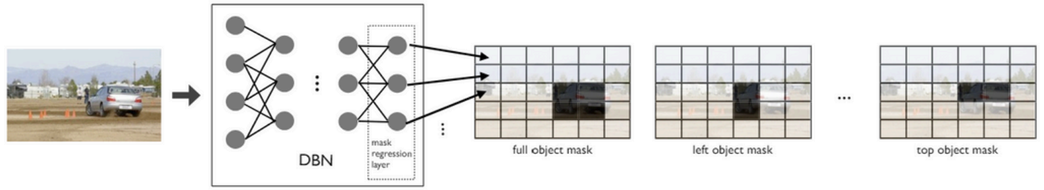


Figure 1: A schematic view of object detection as DNN-based regression.

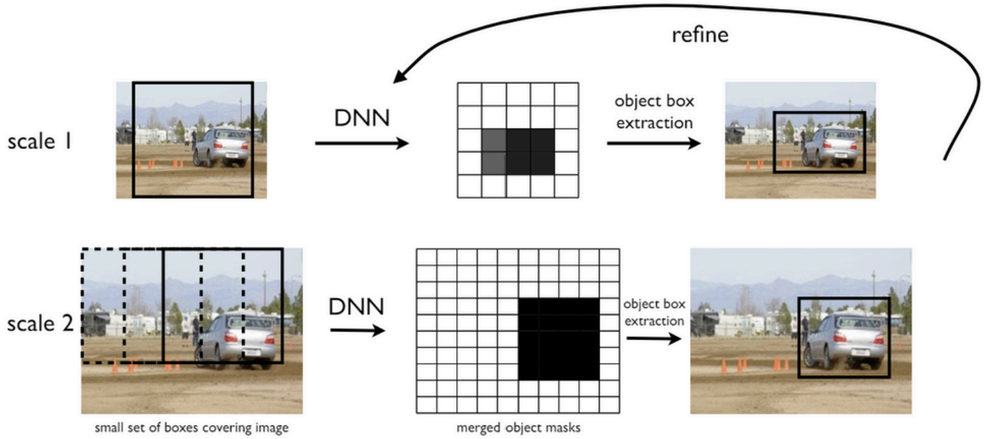
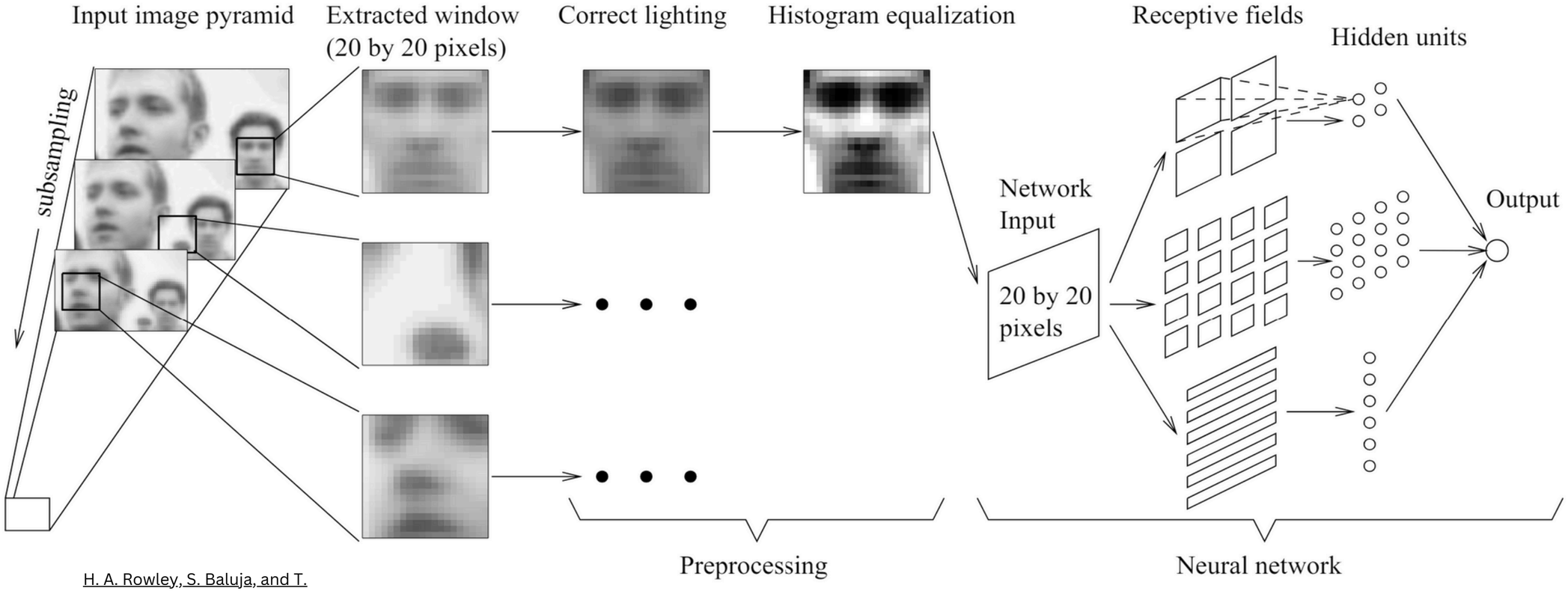


Figure 2: After regressing to object masks across several scales and large image boxes, we perform object box extraction. The obtained boxes are refined by repeating the same procedure on the sub images, cropped via the current object boxes. For brevity, we display only the full object mask, however, we use all five object masks.

Deep Neural Networks for Object Detection
Christian Szegedy Alexander Toshev Dumitru Erhan
Google, Inc.

- Another approach is to build a sliding window detector. This approach has been used for at least 2 decades, typically on constrained object categories such as faces and pedestrians.
- This approach was considered but units high up in the R-CNN's network, which have 5 convolutional layers, have very large receptive fields (195 x 195 pixels) and strides (32 x 32 pixels) in the input image, which makes precise localization within the sliding-window paradigm challenge.

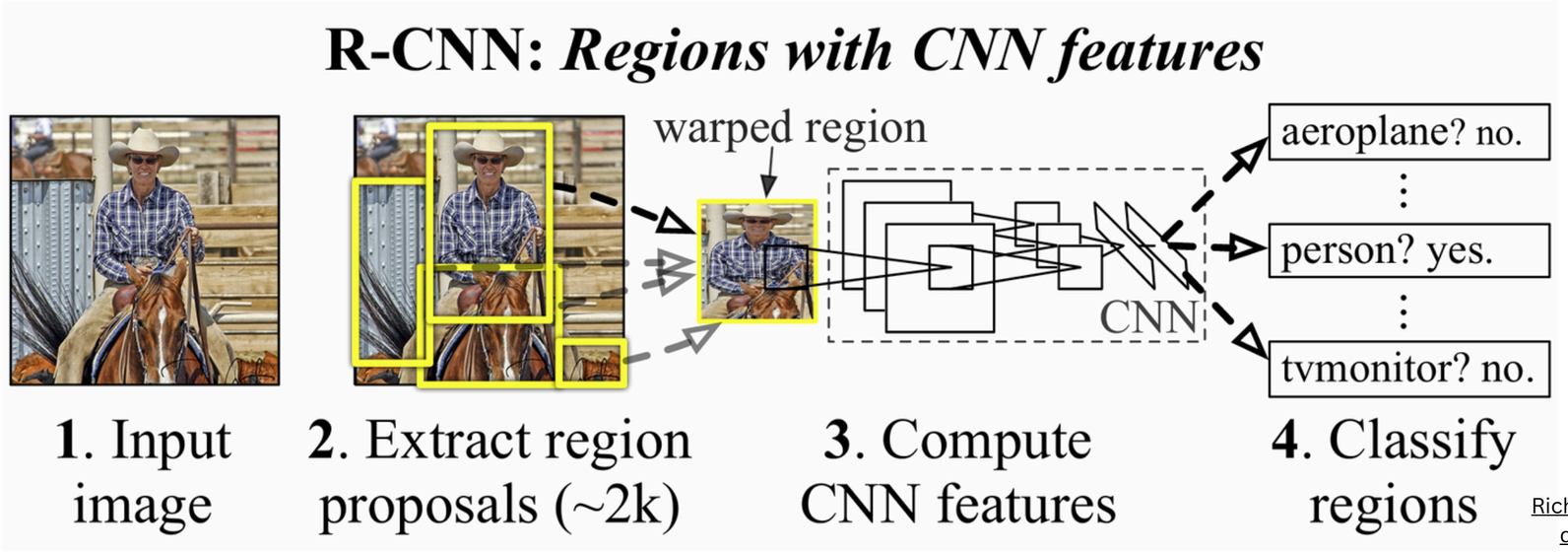


H. A. Rowley, S. Baluja, and T. Kanade. Neural networkbased face detection

Figure 1: The basic algorithm used for face detection.

Approach Used : Recognition Using Regions

- Their method generates around 2000 category-independent region proposals for the input image.
- They use a simple technique (affine image warping) to compute a fixed-size CNN input from each region proposal, regardless of the region's shape.
- Extracts a fixed-length feature vector from each proposal using a CNN.
- Then classifies each region with category-specific linear SVMs.



Rich feature hierarchies for accurate object detection and semantic segmentation
Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik

Now, let’s look at the second problem : Training a high capacity model with only a small quantity of annotated detection data.

- So, the labeled data is scarce and not sufficient for training a large CNN.
- The conventional solution to this problem is to use unsupervised pre-training, followed by supervised fine-tuning.
- Authors of R-CNN show that supervised pre-training on a large auxiliary dataset (ILSVRC), followed by domain specific fine-tuning on a small dataset (PASCAL), is an effective paradigm for learning high-capacity CNNs when data is scarce.

Object Detection with R-CNN

The object detection system consists of 3 modules :-

- The first generates category-independent region proposals. These proposals define the set of candidate detections available to the detector.
- The second module is a large convolutional neural network that extracts a fixed-length feature vector from each region.
- The third module is a set of class specific linear SVMs.

Let’s discuss each of these in detail...

Region Proposals

- A variety of methods are available for generating category independent region proposals. Eg - objectness, selective search, category-independent object proposals, constrained parametric min-cuts (CPMC), multi-scale combinatorial grouping etc.
- R-CNN functions independent of the method chosen for region proposal, therefore all of these methods are viable. However, the authors of the paper chose **Selective Search** as to offer a better comparison with prior detection work.
- Let’s quickly understand how Selective Search works!

Selective Search

- Initial Image Segmentation
  - The algorithm starts by segmenting the image into many small regions using a method like **Felzenszwalb and Huttenlocher's** graph-based image segmentation algorithm.
  - This segmentation aims to create **superpixels**, which are small, perceptually meaningful regions.
- Region Hierarchies
  - The segmented regions are then hierarchically grouped based on their similarities. This is done iteratively to form larger regions.

- First the similarities between all neighbouring regions are calculated.
- The two most similar regions are grouped together, and new similarities are calculated between the resulting region and its neighbours.
- The process of grouping the most similar regions is repeated until the whole image becomes a single region.

Algorithm 1: Hierarchical Grouping Algorithm

Input: (colour) image
Output: Set of object location hypotheses L

Obtain initial regions $R = \{r_1, \cdots, r_n\}$ using Felzenswalb's algorithm
Initialise similarity set $S = \emptyset$
foreach *Neighbouring region pair* (r_i, r_j) **do**
    Calculate similarity $s(r_i, r_j)$
    $S = S \cup s(r_i, r_j)$

while $S \neq \emptyset$ **do**
    Get highest similarity $s(r_i, r_j) = \max(S)$
    Merge corresponding regions $r_t = r_i \cup r_j$
    Remove similarities regarding $r_i : S = S \setminus s(r_i, r_*)$
    Remove similarities regarding $r_j : S = S \setminus s(r_*, r_j)$
    Calculate similarity set S_t between r_t and its neighbours
    $S = S \cup S_t$
    $R = R \cup r_t$

Extract object location boxes L from all regions in R

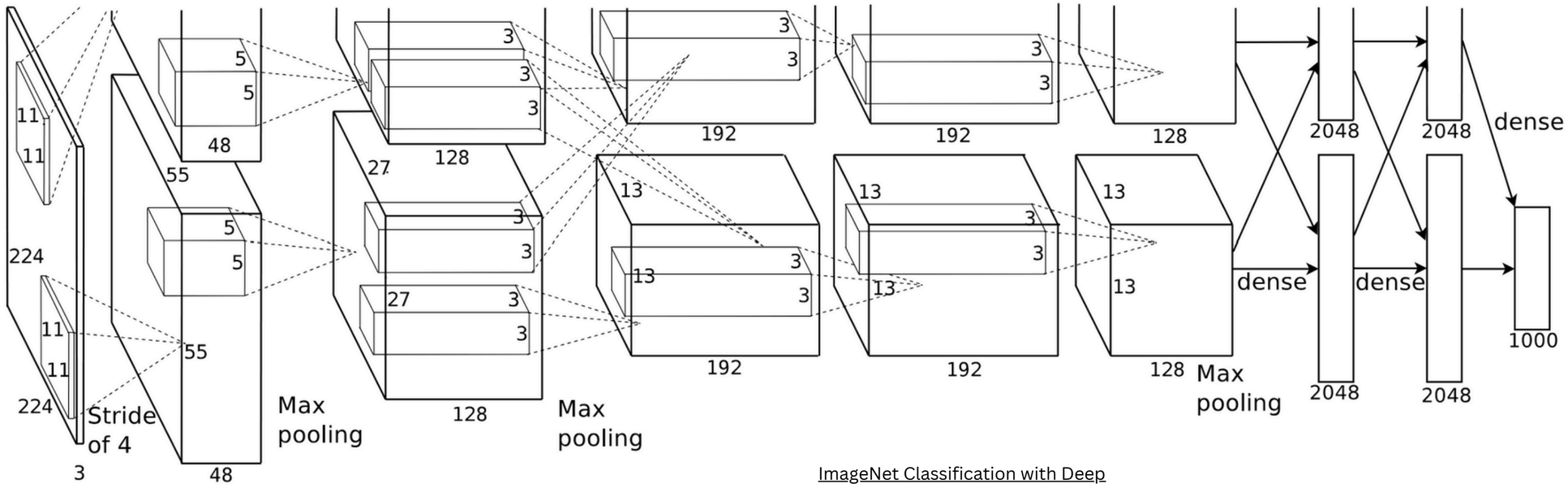
Selective Search for Object Recognition J. R. R. Uijlings · K. E. A. van de Sande · T. Gevers · A. W. M. Smeulders

Efficient Graph-Based Image Segmentation
PEDRO F. FELZENSZWALB

- Similarity Measures - To determine the similarity between regions, Selective Search uses various similarity measures:
  - **Color Similarity**: Measures the similarity in color histograms of two regions.
  - **Texture Similarity**: Uses texture histograms to compare regions.
  - **Size Similarity**: Prefers merging regions of similar sizes.
  - **Shape Compatibility**: Ensures the regions being merged form a compact shape, preventing elongated or irregular shapes.

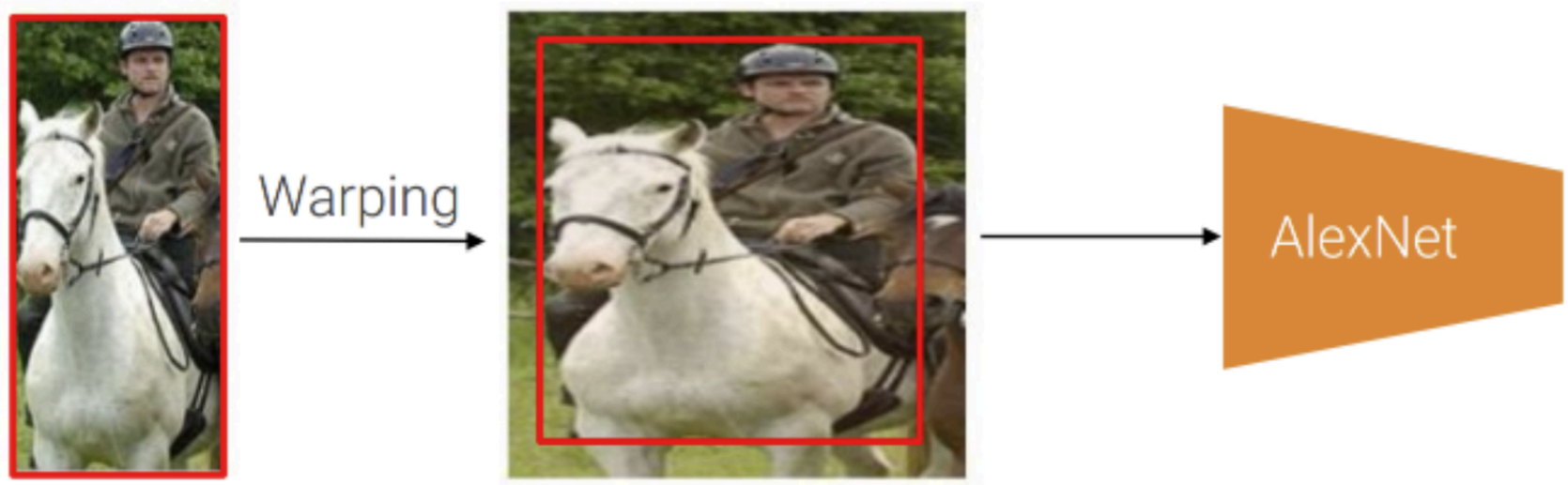
Feature Extraction

- They extract a 4096 dimensional feature vector from each region proposal using a CNN trained on ImageNet (for eg. AlexNet as mentioned in the original paper).
- Features are computed by forward propagating a mean-subtracted 227x227 RGB image (varies based on the chosen CNN architecture) through five convolutional layers and two fully connected layers.



ImageNet Classification with Deep Convolutional Neural Networks
Alex Krizhevsky

- But before we compute features for a region proposal, the image data in that region must be converted to a form that is compatible with the CNN. (For eg. 227 x 227)
- Regardless of the size or or aspect ratio of the candidate region, we warp all pixels in a tight bounding box around it to the required size.
- Prior to warping, we dilate the tight bounding box so that at the warped size there are exactly p pixels of warped image context around the original box (we use p = 16).



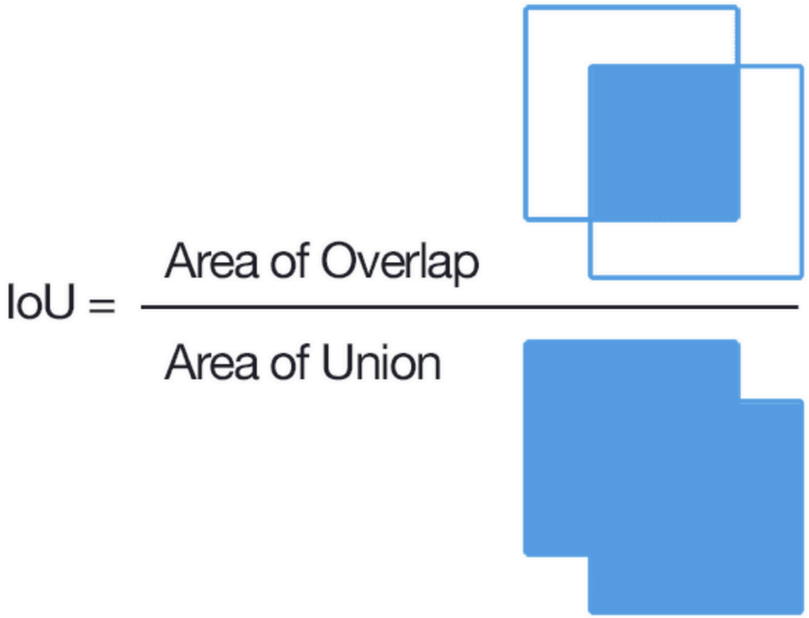
Let’s discuss 2 important concepts before proceeding further into training and inference:-

1. **Intersection over Union**
2. **Non Maximum Suppression**

Intersection over Union (IoU)

- Intersection over Union (IoU) is a metric used to evaluate the accuracy of object detection models, such as object localization and image segmentation.
- It measures the overlap between two bounding boxes: the predicted bounding box and the ground truth bounding box.
- An IoU threshold (e.g., 0.5) is often set to determine if a predicted bounding box is a true positive or a false positive. If IoU > threshold, the prediction is considered accurate.

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



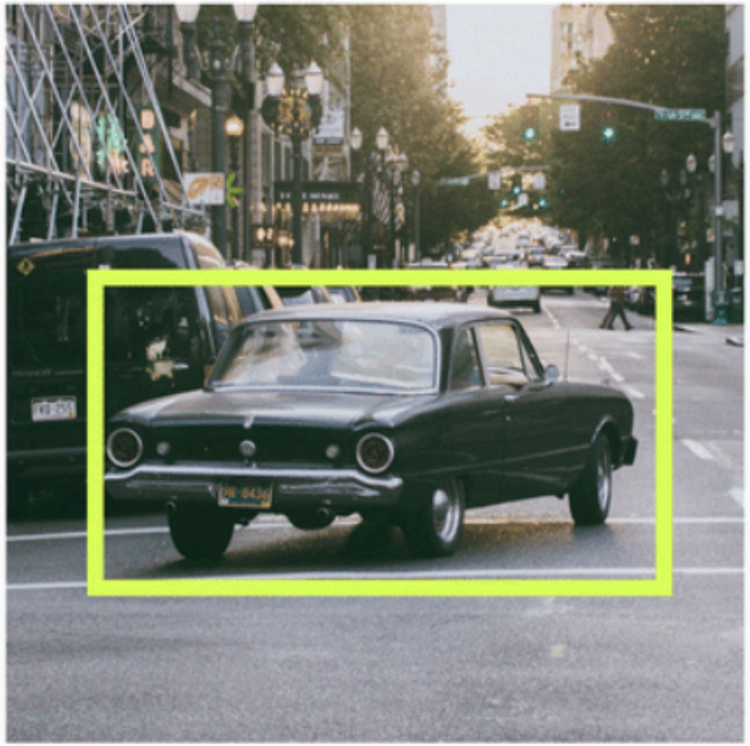
Non-maximum Suppression

- Non-Maximum Suppression (NMS) is a technique used to select the best bounding box among multiple overlapping boxes that predict the same object. It helps to reduce redundant and overlapping bounding boxes and helps ensure that only the most accurate and relevant bounding box is retained for each detected object.
- Steps in NMS :-
    a. Filter out bounding boxes that have a confidence score below a certain threshold.
    b. Sort the remaining bounding boxes based on their confidence scores in descending order.
    c. Start with the highest-scoring bounding box and select it as the best bounding box.
    d. Compute the Intersection over Union (IoU) of the selected box with all other boxes and remove (suppress) all boxes that have an IoU greater than a predefined threshold (e.g., 0.5) with the selected box. These boxes are considered redundant because they overlap significantly with the selected box.
    e. Repeat the process with the next highest-scoring box from the remaining set of boxes until no more boxes remain.

Before non-max suppression

Non-Max Suppression

After non-max suppression



Training

- Supervised pre-training:-
  - The objective is to leverage a large dataset to learn features that can be useful for various tasks.
  - A CNN architecture like AlexNet or VGG is trained on a large scale dataset such as ImageNet.
  - The authors of R-CNN discriminatively pre-trained the CNN on a large auxiliary dataset (ILSVRC2012 classification) using image-level annotations only (bounding box labels are not available for this data).
- Domain-specific fine-tuning:-
  - To adapt the CNN to the new task (detection) and the new domain (warped proposal windows), we continue stochastic gradient descent (SGD) training of the CNN parameters using only warped region proposals.
  - The CNN's ImageNet-specific 1000-way classification layer is replaced with a randomly initialized (N + 1)-way classification layer (where N is the number of object classes, plus 1 for background), and the rest of the CNN architecture is kept unchanged. For eg. for VOC, N = 20.
  - All region proposals with greater than 0.5 IoU overlap with a ground-truth box are treated as positives for that box's class and the rest as negatives.

- Object category classifiers:-
  - Let’s consider training a binary classifier to detect cars (for example). It’s clear that an image region tightly enclosing a car should be a positive example and one without the car should be a negative example.
  - But it’s not clear how to label a region that partially overlaps a car.
  - The authors of R-CNN solve this issue with an IoU overlap threshold, below which regions are defined as negatives. The overlap threshold, 0.3, was selected by a grid search over {0, 0.1,...,0.5} on a validation set.
  - Positive examples are defined simply to be the ground-truth bounding boxes for each class.
  - Proposals that fall in the grey zone (more than 0.3 IoU overlap, but are not ground-truth) are ignored.
  - Once features are extracted and training labels are applied, we optimize one linear SVM per class.
  - Since the training data is too large to fit in memory, the standard hard negative mining method is adopted.

Hard Negative Mining :-

1. Train the SVM classifier using an initial set of positive and negative examples.
2. Use the initially trained SVM to classify all region proposals in the training dataset and identify the false positives (negative examples that were incorrectly classified as positives).
3. Re-train the SVM classifier using the original positive examples and the newly collected hard negative examples.
4. Repeat the process of detecting, collecting hard negatives, and re-training the SVM until the classifier's performance converges

So, to quickly summarize the training:-

- Train CNN on large classification dataset
- Fine tune CNN with resized proposals on classes of detection dataset and background class
  - Proposals with label as Class K - Proposals with IoU >= 0.5 with GT boxes of that class
  - Proposals with label as Class Background - All remaining proposals
- Train a binary classifier (SVM) for each class on the FC layer representation of proposals
  - Positibe Labeled Proposals for class K - GT boxes of that class
  - Negative Labeled Proposals for class K - Proposals with <0.3 IoU with ALL GT boxes of class K
  - Ignore all remaining proposals (with IoU > 0.3 but not GT boxes)

Bounding Box Regression

- After scoring each selective search proposal with a class-specific detection SVM, we predict a new bounding box for the detection using a class-specific bounding box regressor.
- This helps improve localization performance.
- The input to our training algorithm is a set of N training pairs $\{(P^i, G^i)\}_{i=1,\ldots,N}$, where $P^i = (P^i_x, P^i_y, P^i_w, P^i_h)$
- Here, Px and Py specify the pixel coordinates of the center of proposal and Pw and Ph specify the width and height in pixels.

We parameterize the transformation in terms of four functions $d_x(P)$, $d_y(P)$, $d_w(P)$, and $d_h(P)$. The first two specify a scale-invariant translation of the center of P's bounding box, while the second two specify log-space translations of the width and height of P's bounding box. After learning these functions, we can transform an input proposal P into a predicted ground-truth box $\hat{G}$ by applying the transformation

$$\hat{G}_x = P_w d_x(P) + P_x$$
$$\hat{G}_y = P_h d_y(P) + P_y$$
$$\hat{G}_w = P_w \exp(d_w(P))$$
$$\hat{G}_h = P_h \exp(d_h(P)).$$

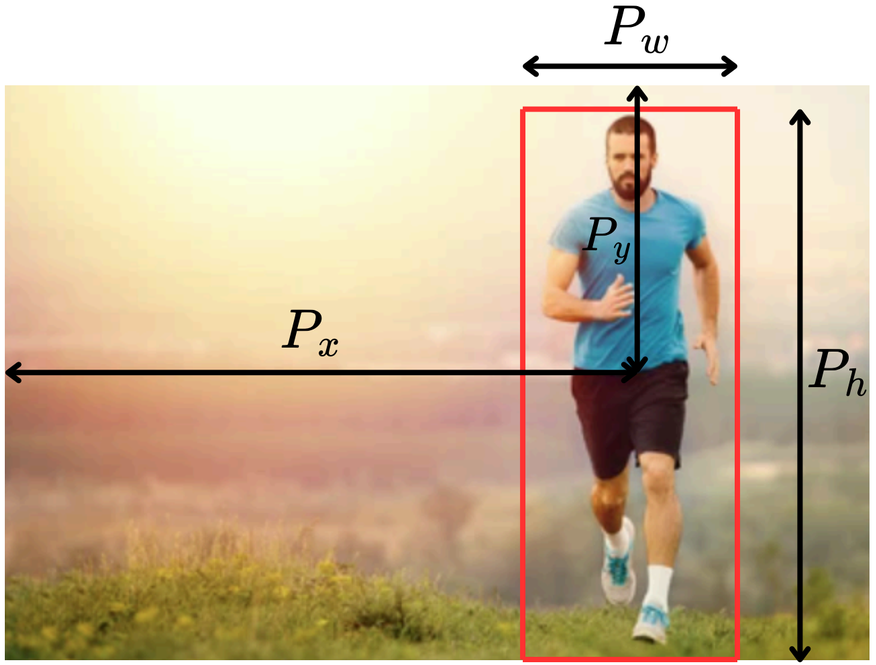
Each function $d_\star(P)$ (where $\star$ is one of x, y, h, w) is modeled as a linear function of the pool_5 features of proposal P, denoted by $\boldsymbol{\phi}_5(P)$. (The dependence of $\boldsymbol{\phi}_5(P)$ on the image data is implicitly assumed.) Thus we have $d_\star(P) = \mathbf{w}_\star^{\mathrm{T}} \boldsymbol{\phi}_5(P)$, where $\mathbf{w}_\star$ is a vector of learnable model parameters. We learn $\mathbf{w}_\star$ by optimizing the regularized least squares objective (ridge regression):

$$\mathbf{w}_\star = \underset{\hat{\mathbf{w}}_\star}{\operatorname{argmin}} \sum_i^N (t_\star^i - \hat{\mathbf{w}}_\star^{\mathrm{T}} \boldsymbol{\phi}_5(P^i))^2 + \lambda \left\| \hat{\mathbf{w}}_\star \right\|^2 .$$

The regression targets $t_\star$ for the training pair (P, G) are defined as

$$t_x = (G_x - P_x)/P_w \tag{6}$$
$$t_y = (G_y - P_y)/P_h \tag{7}$$
$$t_w = \log(G_w/P_w) \tag{8}$$
$$t_h = \log(G_h/P_h). \tag{9}$$

As a standard regularized least squares problem, this can be solved efficiently in closed form.



CNN $pool_5$

FC 1

FC 2

FC 3

FC 4

$$t_x = \frac{G_x - P_x}{P_w}$$

$$t_y = \frac{G_y - P_y}{P_h}$$

$$t_w = log\left(\frac{G_w}{P_w}\right)$$

$$t_h = log\left(\frac{G_h}{P_h}\right)$$

Test Time Detection (Inference)

- At test time, selective search is run on the test image to extract around 2000 region proposals (selective search’s “fast mode” is used in all experiments).
- Each proposal is warped and forward propagated through the CNN in order to compute features.
- Then, for each class, each extracted feature vector is scored using the SVM trained for that class.
- Given all scored regions in an image, a greedy non-maximum suppression is applied (for each class independently) that rejects a region if it has an intersection-over union (IoU) overlap with a higher-scoring selected region larger than a learned threshold.

The choice of CNN matters

VOC 2007 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
R-CNN T-Net	64.2	69.7	50.0	41.9	32.0	62.6	71.0	60.7	32.7	58.5	46.5	56.1	60.6	66.8	54.2	31.5	52.8	48.9	57.9	64.7	54.2
R-CNN T-Net BB	68.1	72.8	56.8	43.0	36.8	66.3	74.2	67.6	34.4	63.5	54.5	61.2	69.1	68.6	58.7	33.4	62.9	51.1	62.5	64.8	58.5
R-CNN O-Net	71.6	73.5	58.1	42.2	39.4	70.7	76.0	74.5	38.7	71.0	56.9	74.5	67.9	69.6	59.3	**35.7**	62.1	64.0	66.5	**71.2**	62.2
R-CNN O-Net BB	**73.4**	**77.0**	**63.4**	**45.4**	**44.6**	**75.1**	**78.1**	**79.8**	**40.5**	**73.7**	**62.2**	**79.4**	**78.1**	**73.1**	**64.2**	35.6	**66.8**	**67.2**	**70.4**	71.1	**66.0**

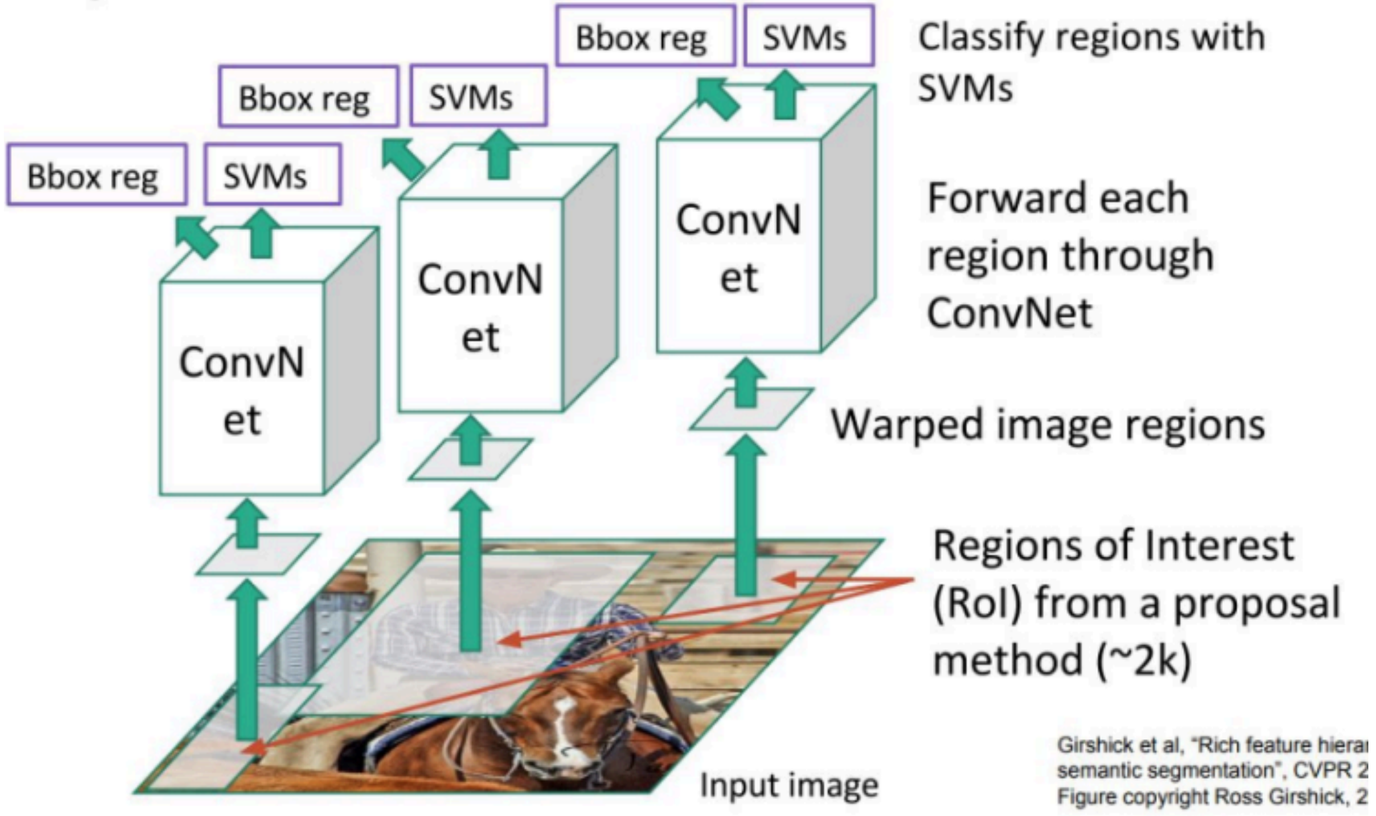
Table 3: Detection average precision (%) on VOC 2007 test for two different CNN architectures. The first two rows are results from Table 2 using Krizhevsky et al.'s architecture (T-Net). Rows three and four use the recently proposed 16-layer architecture from Simonyan and Zisserman (O-Net) [43].

Results with and without fine-tuning, enhancement with bounding-box regression and more

VOC 2007 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
R-CNN $pool_5$	51.8	60.2	36.4	27.8	23.2	52.8	60.6	49.2	18.3	47.8	44.3	40.8	56.6	58.7	42.4	23.4	46.1	36.7	51.3	55.7	44.2
R-CNN fc_6	59.3	61.8	43.1	34.0	25.1	53.1	60.6	52.8	21.7	47.8	42.7	47.8	52.5	58.5	44.6	25.6	48.3	34.0	53.1	58.0	46.2
R-CNN fc_7	57.6	57.9	38.5	31.8	23.7	51.2	58.9	51.4	20.0	50.5	40.9	46.0	51.6	55.9	43.3	23.3	48.1	35.3	51.0	57.4	44.7
R-CNN FT $pool_5$	58.2	63.3	37.9	27.6	26.1	54.1	66.9	51.4	26.7	55.5	43.4	43.1	57.7	59.0	45.8	28.1	50.8	40.6	53.1	56.4	47.3
R-CNN FT fc_6	63.5	66.0	47.9	37.7	29.9	62.5	70.2	60.2	32.0	57.9	47.0	53.5	60.1	64.2	52.2	31.3	55.0	50.0	57.7	63.0	53.1
R-CNN FT fc_7	64.2	69.7	50.0	41.9	32.0	62.6	71.0	60.7	32.7	58.5	46.5	56.1	60.6	66.8	54.2	31.5	52.8	48.9	57.9	64.7	54.2
R-CNN FT fc_7 BB	**68.1**	**72.8**	**56.8**	**43.0**	**36.8**	**66.3**	**74.2**	**67.6**	**34.4**	**63.5**	**54.5**	**61.2**	**69.1**	**68.6**	**58.7**	**33.4**	**62.9**	**51.1**	**62.5**	**64.8**	**58.5**
DPM v5 [20]	33.2	60.3	10.2	16.1	27.3	54.3	58.2	23.0	20.0	24.1	26.7	12.7	58.1	48.2	43.2	12.0	21.1	36.1	46.0	43.5	33.7
DPM ST [28]	23.8	58.2	10.5	8.5	27.1	50.4	52.0	7.3	19.2	22.8	18.1	8.0	55.9	44.8	32.4	13.3	15.9	22.8	46.2	44.9	29.1
DPM HSC [31]	32.2	58.3	11.5	16.3	30.6	49.9	54.8	23.5	21.5	27.7	34.0	13.7	58.1	51.6	39.9	12.4	23.5	34.4	47.4	45.2	34.3

Table 2: Detection average precision (%) on VOC 2007 test. Rows 1-3 show R-CNN performance without fine-tuning. Rows 4-6 show results for the CNN pre-trained on ILSVRC 2012 and then fine-tuned (FT) on VOC 2007 trainval. Row 7 includes a simple bounding-box regression (BB) stage that reduces localization errors (Section C). Rows 8-10 present DPM methods as a strong baseline. The first uses only HOG, while the next two use different feature learning approaches to augment or replace HOG.

Limitations



- Training is expensive and computationally heavy because of selective search and lack of shared computation.
- Each proposed region is processed separately which makes it inefficient for real time applications.

**Image from lecture slides by Jiageng Zhang, Jingyao Zhan, Yunhan Ma*

Intro to Fast R-CNN

Notable drawbacks of R-CNN :-

1. **Training is a multi-stage pipeline.** R-CNN first fine tunes a ConvNet on object proposals using log loss. Then, it fits SVMs to ConvNet features. These SVMs act as object detectors, replacing the softmax classifier learnt by fine-tuning. In the third training stage, bounding-box regressors are learned.
2. **Training is expensive in space and time.** For SVM and bounding-box regressor training, features are extracted from each object proposal in each image and written to disk. With very deep networks, such as VGG16, this process takes 2.5 GPU-days for the 5k images of the VOC-2007 train-val set. These features require hundreds of gigabytes of storage.
3. **Object detection is slow.** At test-time, features are extracted from each object proposal in each test image. Detection with VGG16 takes 47s / image (on a GPU).

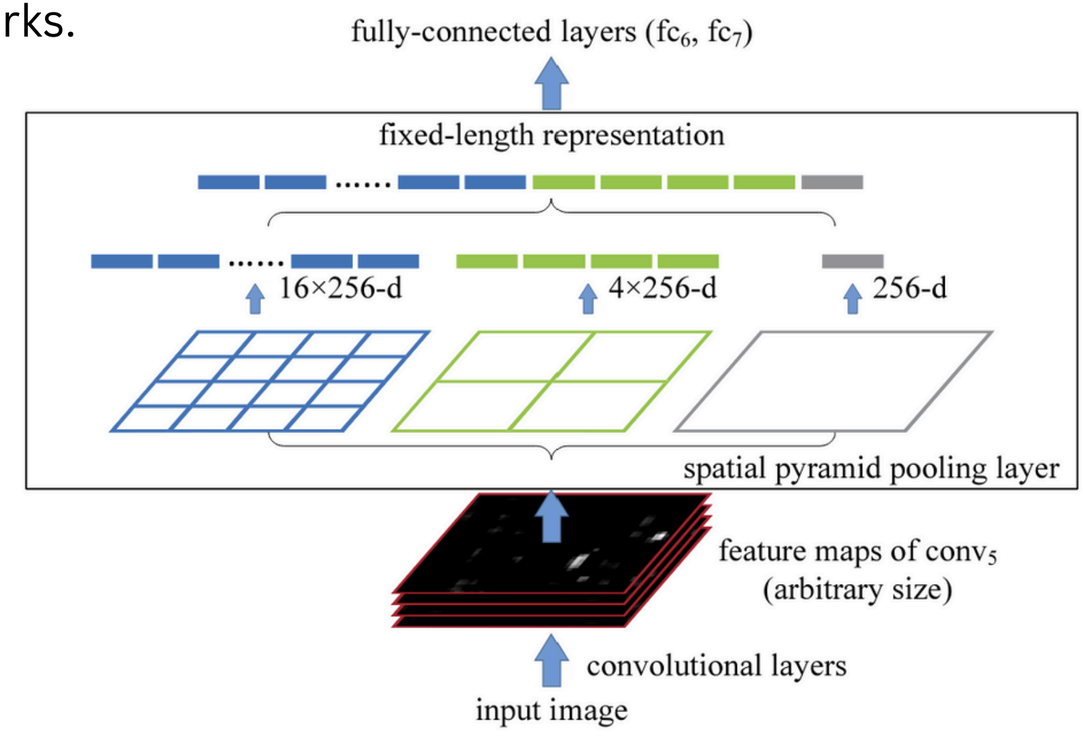
Spatial Pyramid Pooling networks (SPPnets)

- R-CNN is slow because it performs a ConvNet forward pass for each object proposal, without sharing computation.
- SPPnets were proposed to speed up R-CNN by sharing computation.
- The SPPnet method computes a convolutional feature map for the entire input image and then classifies each object proposal using a feature vector extracted from the shared feature map.
- Features are extracted for a proposal by max pooling the portion of the feature map inside the proposal into a fixed-size output (e.g., 6 x 6). Multiple output sizes are pooled and then concatenated as in spatial pyramid pooling.
- **SPPnet accelerates R-CNN by 10 to 100x at test time. Training time is also reduced by 3x** due to faster proposal feature extraction.

Notable drawbacks of SPPnet :-

1. Like R-CNN, training is a multi-stage pipeline that involves extracting features, fine-tuning a network with log loss, training SVMs, and finally fitting bounding-box regressors. Features are also written to disk.
2. But unlike R-CNN, the fine-tuning algorithm proposed in SPPnet cannot update the convolutional layers that precede the spatial pyramid pooling. Unsurprisingly, this limitation (fixed convolutional layers) limits the accuracy of very deep networks.

A network structure with a spatial pyramid pooling layer. Here 256 is the filter number of the conv5 layer, and conv5 is the last convolutional layer.



Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun

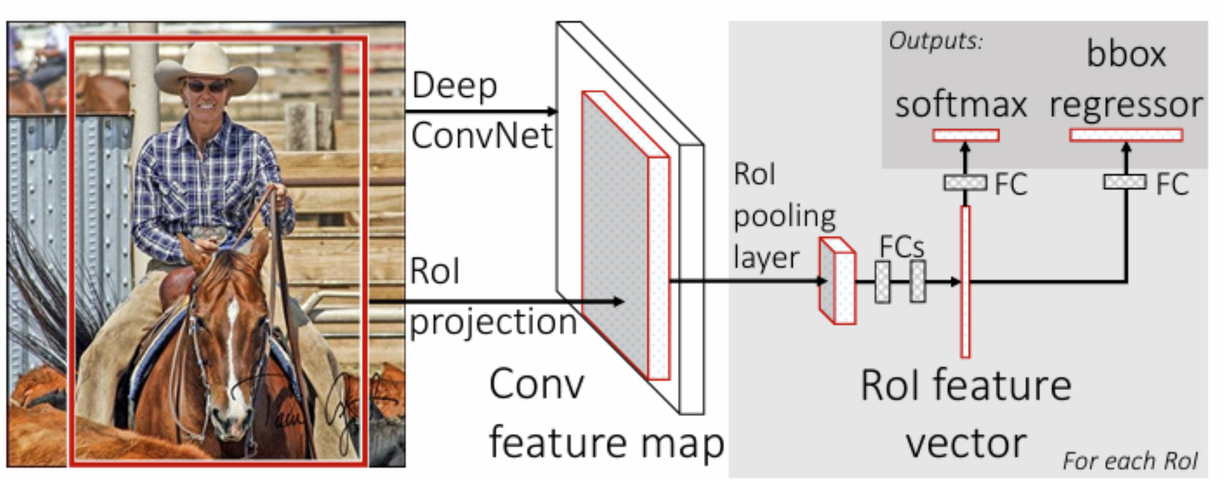
Contributions/Advantages of Fast R-CNN

1. Higher detection quality (mAP) than R-CNN, SPPnet
2. Training is single-stage, using multi-task loss
3. Training can update all network layers
4. No disk storage is required for feature caching

Fast R-CNN
Ross Girshick

Fast R-CNN Architecture

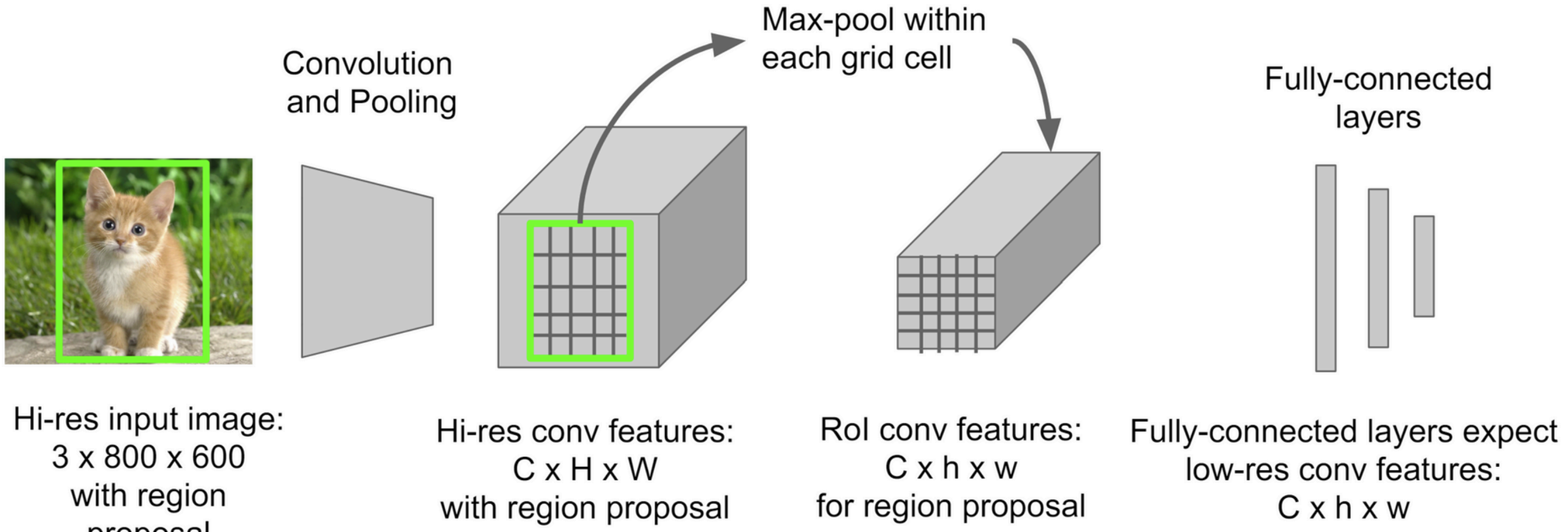
- Fast R-CNN network takes as input
  a. An entire image
  b. A set of object proposals.
- The network first processes the whole image with several convolutional (conv) and max pooling layers to produce a conv feature map.
- Then, for each object proposal a region of interest (RoI) pooling layer extracts a fixed-length feature vector from the feature map.
- Each feature vector is fed into a sequence of fully connected (fc) layers that finally branch into two sibling output layers:
  - One that produces softmax probability estimates over K object classes plus a catch-all “background” class
  - And another layer that outputs four real-valued numbers for each of the K object classes. Each set of 4 values encodes refined bounding-box positions for one of the K classes.



- An input image and multiple regions of interest (ROIs) are input into a fully convolutional network.
- Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs).
- The network has two output vectors per RoI:
  - softmax probabilities
  - per-class bounding-box regression offsets.
- The architecture is trained end-to-end with a multi-task loss.

RoI Pooling Layer

- The RoI pooling layer uses max pooling to convert the features inside any valid region of interest into a small feature map with a fixed spatial extent of H x W (e.g., 7 x 7), where H and W are layer hyper-parameters that are independent of any particular RoI.
- In Fast R-CNN, an RoI is a rectangular window into a conv feature map. Each RoI is defined by a four-tuple (r,c,h,w) that specifies its top-left corner (r,c) and its height and width (h,w).
- RoI max pooling works by dividing the h x w RoI window into an H x W grid of sub-windows of approximate size h/H x w/W.
- We then max-pool the values in each sub-window into the corresponding output grid cell. Pooling is applied independently to each feature map channel, as in standard max pooling.



The main job of the RoI pooling layer is to convert the region proposals in the convolutional feature map to a fixed dimension (h x w) which can be flattened and fed as input to the FC layers for further classification and regression.

Slide credits : cs231n lecture slides

RoI Pooling Layer Implementation

Let’s understand with an example:-

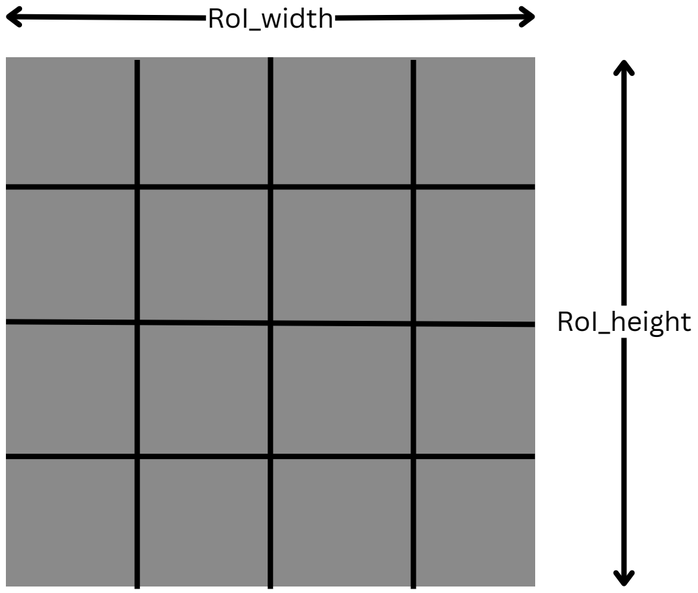
- Suppose our input image has dimensions 400 x 400 x 3 and the CNN downscales it by a factor of 40, and we obtain a feature map with dimensions 10 x 10 x 512.
- Since the image has been downscaled, we might be unable to map the exact region proposals on the feature map. This is because region proposals were defined on a 400 x 400 image, and unless the coordinates are a multiple of 40, they can’t be accurately mapped to the 10 x 10 feature map.
- In order to extract feature map regions for such cases, we quantize the location of proposal in the feature map by rounding.
- This would lead to some regions which were part of the region proposal being excluded and some regions which were not part of the region proposal be included.
- Now, we have obtained feature maps for each region proposal. But these might be of different sizes. We need same size feature maps to feed into FC layers!

- Suppose we require a 2 x 2 feature map for each region proposal to feed into the FC layers.
- As mentioned earlier we'll divide each region proposal's feature map into 4 parts (2 x 2) and apply max pool separately on each of these 4 parts to obtain a 2 x 2 feature map for that region proposal.
- These fixed size feature maps can now be fed into the FC layers.
- But wait, dividing a region proposal's feature map into some n parts (in this case, 4) might not always be that trivial.
- Assume after approximating, you obtain a feature map of dimensions 8 x 7 for a region proposal. Now how do you divide this into 4 parts?
- Therefore, we need a common approach to divide feature maps (H x W) into (h x w) parts. (h x w) is the input size required by the FC layers.
- The authors of Fast R-CNN use the approach mentioned in another paper: SPPnet

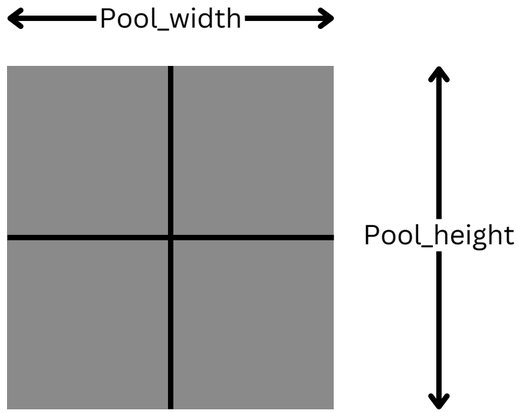
Implementation of Pooling Bins.

We use the following implementation to handle all bins when applying the network. Denote the width and height of the conv_5 feature maps (can be the full image or a window) as w and h. For a pyramid level with $n \times n$ bins, the (i, j)-th bin is in the range of $[\lfloor \frac{i-1}{n} w \rfloor, \lceil \frac{i}{n} w \rceil] \times [\lfloor \frac{j-1}{n} h \rfloor, \lceil \frac{j}{n} h \rceil]$. Intuitively,

Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun



bin_w = Roi_width / Pool_width
bin_h = Roi_height / Pool_height



So, for a given (i,j) cell of the pooled feature map. The area of the RoI feature map to be considered is given by :-

w_start = floor(i * bin_w)
h_start = floor(j * bin_h)

w_end = ceil(i * bin_w)
h_end = ceil(j * bin_h)

Here, (w_start, h_start) define the bottom left coordinates and (w_end, h_end) define the up rightmost coordinates.

Initializing from pre-trained networks

When a pre-trained network initializes a Fast R-CNN network, it undergoes three transformations:-

1. First, the last max pooling layer is replaced by a RoI pooling layer that is configured by setting H and W to be compatible with the net's first fully connected layer (e.g., H = W = 7 for VGG16).
2. Second, the network's last fully connected layer and soft max (which were trained for 1000-way ImageNet classification) are replaced with the two sibling layers described earlier (a fully connected layer and softmax over K +1 categories and category-specific bounding-box regressors).
3. Third, the network is modified to take two data inputs: an image (or an image pyramid, encoded as a list of images) and a list of RoI proposals in those images.

Fine tuning for Detection

Training all network weights with back-propagation is an important capability of Fast R-CNN.

Why is SPPnet unable to update weights below the spatial pyramid pooling layer?

- The root cause is that back-propagation through the SPP layer is highly inefficient when each training sample (i.e. RoI) comes from a different image, which is exactly how R-CNN and SPPnet networks are trained.
- The inefficiency stems from the fact that each RoI may have a very large receptive field, often spanning the entire input image.
- Since the forward pass must process the entire receptive field, the training inputs are large (often the entire image).

- Fast R-CNN takes advantage of feature sharing during training.
- In Fast R CNN training, stochastic gradient descent (SGD) mini batches are sampled hierarchically, first by sampling N images and then by sampling R/N RoIs from each image.
- Critically, RoIs from the same image share computation and memory in the forward and backward passes.
- Making N small decreases mini-batch computation. For example, when using N = 2 and R = 128, the proposed training scheme is roughly 64 faster than sampling one RoI from 128 different images (i.e., the R-CNN and SPPnet strategy).
- In addition to hierarchical sampling, Fast R-CNN uses a streamlined training process with one fine-tuning stage that jointly optimizes a softmax classifier and bounding-box regressors, rather than training a softmax classifier, SVMs, and regressors in three separate stages

Training components of Fast R-CNN

1. Multi-task Loss
2. Mini-batch Sampling
3. Back-propagation through RoI Pooling Layers
4. SGD hyper-parameters

Multi task Loss

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1]L_{\text{loc}}(t^u, v),$$

- Combination of classification and localization loss catering to both the layers. Lamda balances between the two types of losses. (In implementations, lamda is simply taken to be 1)
- Classification loss is the cross-entropy loss using the K=1 output probabilities from the classification layer and the ground truth class u for a given proposal.
- The localization part is the same as in the case of R-CNN but instead of using MSE, we use L1 loss.

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}(t_i^u - v_i),$$

in which

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

Mini Batch Sampling

- During fine-tuning, each SGD mini-batch is constructed from N = 2 images, chosen uniformly at random (as is common practice, we actually iterate over permutations of the dataset).
- We use mini-batches of size R = 128, sampling 64 RoIs from each image. As in [9], we take 25% of the RoIs from object proposals that have intersection over union (IoU) overlap with a ground truth bounding box of at least 05.
- These RoIs comprise the examples labeled with a foreground object class, i.e. u >= 1.
- The remaining RoIs are sampled from object proposals that have a maximum IoU with ground truth in the interval [0.1,0.5).
- These are the background examples and are labeled with u = 0. The lower threshold of 0.1 appears to act as a heuristic for hard example mining.
- During training, images are horizontally flipped with probability 05. No other data augmentation is used.

Back-propagation through RoI pooling layers

Back propagation routes derivatives through the RoI pooling layer.

- For clarity, we assume only one image per mini-batch (N =1), though the extension to N > 1 is straightforward because the forward pass treats all images independently.

Let $x_i \in \mathbb{R}$ be the i-th activation input into the RoI pooling layer and let y_{rj} be the layer's j-th output from the r-th RoI. The RoI pooling layer computes $y_{rj} = x_{i^*(r,j)}$, in which $i^*(r,j) = \text{argmax}_{i' \in \mathcal{R}(r,j)} x_{i'}$. $\mathcal{R}(r,j)$ is the index set of inputs in the sub-window over which the output unit y_{rj} max pools. A single x_i may be assigned to several different outputs y_{rj}.

The RoI pooling layer's backwards function computes partial derivative of the loss function with respect to each input variable x_i by following the argmax switches:

$$\frac{\partial L}{\partial x_i} = \sum_r \sum_j [i = i^*(r,j)] \frac{\partial L}{\partial y_{rj}}.$$

Scale Invariance

Two ways of achieving scale invariant object detection are explored.

1. “Brute force” Learning -
    - Each image is processed at a pre-defined pixel size during both training and testing.
    - The network must directly learn scale-invariant object detection from the training data.
2. Using image pyramids -
    - The image pyramid is used to approximately scale-normalize each object proposal.
    - During multi-scale training, we randomly sample a pyramid scale each time an image is sampled.

Fast R-CNN Detection

- Assuming object proposals are pre-computed, the network takes as input an image (or an image pyramid, encoded as a list of images) and a list of R object proposals to score. At test time, R is typically around 2000.
- When using an image pyramid, each RoI is assigned to the scale such that the scaled RoI is closest to 224^2 pixels in area.
- For each test RoI r, the forward pass outputs a class posterior probability distribution p and a set of predicted bounding-box offsets relative to r (each of the K classes gets its own refined bounding-box prediction).
- We assign a detection confidence to r for each object class k using the estimated probability.
- We then perform non-maximum suppression independently for each class using the algorithm and settings from R-CNN

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
SPPnet BB [11]†	07 \ diff	73.9	72.3	62.5	51.5	44.4	74.4	73.0	74.4	42.3	73.6	57.7	70.3	74.6	74.3	54.2	34.0	56.4	56.4	67.9	73.5	63.1
R-CNN BB [10]	07	73.4	77.0	63.4	45.4	**44.6**	75.1	78.1	79.8	40.5	73.7	62.2	79.4	78.1	73.1	64.2	**35.6**	66.8	67.2	70.4	**71.1**	66.0
FRCN [ours]	07	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8	66.9
FRCN [ours]	07 \ diff	74.6	**79.0**	68.6	57.0	39.3	79.5	**78.6**	81.9	**48.0**	74.0	67.4	80.5	80.7	74.1	69.6	31.8	67.1	68.4	75.3	65.5	68.1
FRCN [ours]	07+12	**77.0**	78.1	**69.3**	**59.4**	38.3	**81.6**	**78.6**	**86.7**	42.8	**78.8**	**68.9**	**84.7**	**82.0**	**76.6**	**69.9**	31.8	**70.1**	**74.8**	**80.4**	70.4	**70.0**

Table 1. **VOC 2007 test** detection average precision (%). All methods use VGG16. Training set key: **07**: VOC07 trainval, **07 \ diff**: **07** without "difficult" examples, **07+12**: union of **07** and VOC12 trainval. †SPPnet results were prepared by the authors of [11].

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	77.7	73.8	62.3	48.8	45.4	67.3	67.0	80.3	41.3	70.8	49.7	79.5	74.7	78.6	64.5	36.0	69.9	55.7	70.4	61.7	63.8
R-CNN BB [10]	12	79.3	72.4	63.1	44.0	44.4	64.6	66.3	84.9	38.8	67.3	48.4	82.3	75.0	76.7	65.7	35.8	66.2	54.8	69.1	58.8	62.9
SegDeepM	12+seg	**82.3**	75.2	67.1	50.7	**49.8**	71.1	69.6	88.2	42.5	71.2	50.0	85.7	76.6	81.8	69.3	**41.5**	**71.9**	62.2	73.2	**64.6**	67.2
FRCN [ours]	12	80.1	74.4	67.7	49.4	41.4	74.2	68.8	87.8	41.9	70.1	50.2	86.1	77.3	81.1	70.4	33.3	67.0	63.3	77.2	60.0	66.1
FRCN [ours]	07++12	82.0	**77.8**	**71.6**	**55.3**	42.4	**77.3**	**71.7**	**89.3**	**44.5**	**72.1**	**53.7**	**87.7**	**80.0**	**82.5**	**72.7**	36.6	68.7	**65.4**	**81.1**	62.7	**68.8**

Table 2. **VOC 2010 test** detection average precision (%). BabyLearning uses a network based on [17]. All other methods use VGG16. Training set key: **12**: VOC12 trainval, **Prop.**: proprietary dataset, **12+seg**: **12** with segmentation annotations, **07++12**: union of VOC07 trainval, VOC07 test, and VOC12 trainval.

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6	63.2
NUS_NIN_c2000	Unk.	80.2	73.8	61.9	43.7	**43.0**	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	**38.6**	**68.3**	58.0	68.7	63.3	63.8
R-CNN BB [10]	12	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3	62.4
FRCN [ours]	12	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7	65.7
FRCN [ours]	07++12	**82.3**	**78.4**	**70.8**	**52.3**	38.7	**77.8**	**71.6**	**89.3**	**44.2**	**73.0**	**55.0**	**87.5**	**80.5**	**80.8**	**72.0**	35.1	**68.3**	**65.7**	**80.4**	**64.2**	**68.4**

Table 3. **VOC 2012 test** detection average precision (%). BabyLearning and NUS_NIN_c2000 use networks based on [17]. All other methods use VGG16. Training set key: see Table 2, **Unk.**: unknown.

	Fast R-CNN			R-CNN			SPPnet
	S	**M**	**L**	**S**	**M**	**L**	†**L**
train time (h)	**1.2**	2.0	9.5	22	28	84	25
train speedup	**18.3**×	14.0×	8.8×	1×	1×	1×	3.4×
test rate (s/im)	0.10	0.15	0.32	9.8	12.1	47.0	2.3
▷ with SVD	**0.06**	0.08	0.22	-	-	-	-
test speedup	98×	80×	146×	1×	1×	1×	20×
▷ with SVD	169×	150×	**213**×	-	-	-	-
VOC07 mAP	57.1	59.2	**66.9**	58.5	60.2	66.0	63.1
▷ with SVD	56.5	58.7	66.6	-	-	-	-

Table 4. Runtime comparison between the same models in Fast R-CNN, R-CNN, and SPPnet. Fast R-CNN uses single-scale mode. SPPnet uses the five scales specified in [11]. †Timing provided by the authors of [11]. Times were measured on an Nvidia K40 GPU.

Intro to Faster R-CNN

Problem with Fast R-CNN

- Fast R-CNN achieves near real-time rates using very deep networks when ignoring the time spent on region proposals. Apparently, proposals are the test-time computational bottleneck in state-of-the-art detection systems.
- Region proposal methods typically rely on inexpensive features and economical inference schemes. Selective Search is one of the most popular methods. Yet when compared to efficient detection networks, Selective Search is slower, at 2 seconds per image in a CPU implementation.
- Fast region-based CNNs take advantage of GPUs, while the region proposal methods used in research are implemented on the CPU, making such runtime comparisons inequitable.

- The authors of Faster R-CNN show that an algorithmic change— computing proposals with a deep convolutional neural network—leads to an elegant and effective solution where proposal computation is nearly cost-free given the detection network's computation.
- They observe that the convolutional feature maps used by region-based detectors, like Fast R CNN, can also be used for generating region proposals.
- They introduce novel Region Proposal Networks (RPNs) that share convolutional layers with state-of-the-art object detection networks.
- RPNs are designed to efficiently predict region proposals with a wide range of scales and aspect ratios.
- In contrast to prevalent methods that use pyramids of images or pyramids of filters, the authors introduce novel "anchor" boxes that serve as references at multiple scales and aspect ratios.
- To unify RPNs with Fast R-CNN object detection networks, the authors propose a training scheme that alternates between fine-tuning for the region proposal task and then fine-tuning for object detection while keeping the proposals fixed.

Faster R-CNN Architecture

Faster R-CNN is composed of 2 modules:-

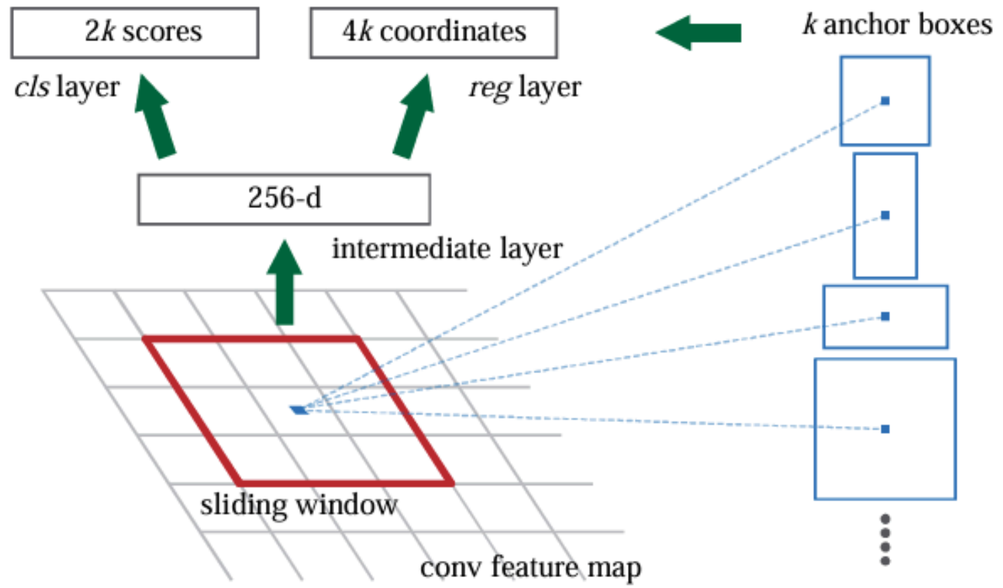
1. The first module is a deep fully convolutional network that proposes regions.
2. And the second module is the Fast R-CNN detector that uses the proposed regions.

Let’s understand the first module i.e. the Region Proposal Network

Region Proposal Networks (RPN)

- A Region Proposal Network (RPN) takes an image (of any size) as input and outputs a set of rectangular object proposals, each with an objectness score.
- This process is modeled with a fully convolutional neural network. Because our ultimate goal is to share computation with a Fast R-CNN object detection network, we assume that both nets share a common set of convolutional layers.
- In the experiments, the authors investigate the Zeiler and Fergus model (ZF), which has 5 shareable convolutional layers, and the Simonyan and Zisserman model (VGG-16), which has 13 shareable convolutional layers.
- To generate region proposals, we slide a small network over the convolutional feature map output by the last shared convolutional layer.
- This small network takes as input an n x n spatial window of the input convolutional feature map. Each sliding window is mapped to a lower-dimensional feature (256-d for ZF and 512-d for VGG, with ReLU following).
- This feature is fed into two sibling fully connected layers—a box-regression layer (reg) and a box-classification layer (cls).

- Note that because the mini-network operates in a sliding-window fashion, the fully-connected layers are shared across all spatial locations. This architecture is naturally implemented with an n x n convolutional layer followed by two sibling 1 x 1 convolutional layers (for reg and cls, respectively).



The Region Proposal Network

Anchors

- At each sliding-window location, we simultaneously predict multiple region proposals, where the number of maximum possible proposals for each location is denoted as k.
- So the reg layer has 4k outputs encoding the coordinates of k boxes, and the cls layer outputs 2k scores that estimate the probability of object or not object for each proposal.
- The k proposals are parameterized relative to k reference boxes, which we call anchors.
- An anchor is centered at the sliding window in question, and is associated with a scale and aspect ratio.
- By default we use 3 scales and 3 aspect ratios, yielding k = 9 anchors at each sliding position. For a convolutional feature map of a size W x H (typically 2,400), there are W x H x k anchors in total.

Anchors are translation invariant

- If one translates an object in an image, the proposal should translate, and the same function should be able to predict the proposal in either location.
- This translation-invariant property is guaranteed by this method.

Multi-Scale Anchors as Regression References

- There have been two popular ways for multi-scale predictions.
  - The first way is based on image/feature pyramids. The images are resized at multiple scales, and feature maps are computed for each scale.
  - The second way is to use sliding windows of multiple scales (and/or aspect ratios) on the feature maps. This can be thought of as a “pyramid of filters”.

- As a comparison, the anchor-based method is built on a pyramid of anchors, which is more cost-efficient.
- This method classifies and regresses bounding boxes with reference to anchor boxes of multiple scales and aspect ratios. It only relies on images and feature maps of a single scale and uses filters (sliding windows on the feature map) of a single size.
- Because of this multi-scale design based on anchors, we can simply use the convolutional features computed on a single-scale image, as is also done by the Fast R-CNN detector.

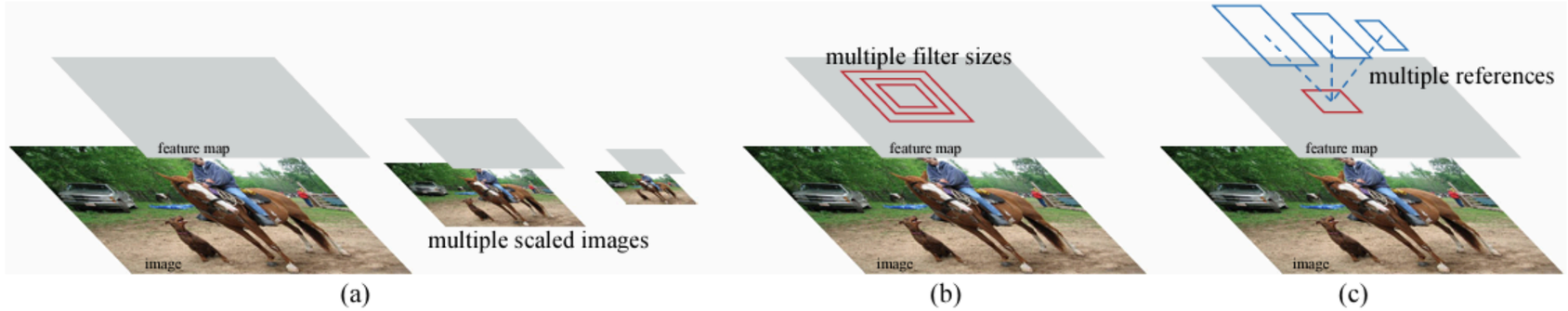


Figure 1: Different schemes for addressing multiple scales and sizes. (a) Pyramids of images and feature maps are built, and the classifier is run at all scales. (b) Pyramids of filters with multiple scales/sizes are run on the feature map. (c) We use pyramids of reference boxes in the regression functions.

Loss Function

- For training RPNs, we assign a binary class label (of being an object or not) to each anchor.
- We assign a positive label to two kinds of anchors:
  - The anchor/anchors with the highest Intersection-over Union (IoU) overlap with a ground-truth box.
  - Or an anchor that has an IoU overlap higher than 0.7 with any ground-truth box. Note that a single ground-truth box may assign positive labels to multiple anchors.
- Usually the second condition is sufficient to determine the positive samples; but we still adopt the first condition for the reason that in some rare cases the second condition may find no positive sample.
- We assign a negative label to a non-positive anchor if its IoU ratio is lower than 0.3 for all ground-truth boxes.
- Anchors that are neither positive nor negative do not contribute to the training objective.

- With these definitions, we minimize an objective function following the multi-task loss in Fast R-CNN.
- The loss function for an image is defined as:

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*).$$

- Here, i is the index of an anchor in a mini-batch and p_i is the predicted probability of anchor i being an object. The ground-truth label p_i is 1 if the anchor is positive, and is 0 if the anchor is negative.
- t_i is a vector representing the 4 parameterized coordinates of the predicted bounding box, and $t_i^{\times}$ is that of the ground-truth box associated with a positive anchor.
- The classification loss L_{cls} is log loss over two classes (object vs. not object).
- For the regression loss, we use $L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$. Here, R is the robust loss function (smooth L1) from Fast R-CNN.
- The term $p_i^{\times} L_{reg}$ means the regression loss is activated only for positive anchors.
- The two terms are normalized by N_{cls} and N_{reg} and weighted by a balancing parameter λ.

- The parametrization for bounding box regression is same as that in R-CNN

$$t_{\mathrm{x}} = (x - x_{\mathrm{a}})/w_{\mathrm{a}}, \quad t_{\mathrm{y}} = (y - y_{\mathrm{a}})/h_{\mathrm{a}},$$
$$t_{\mathrm{w}} = \log(w/w_{\mathrm{a}}), \quad t_{\mathrm{h}} = \log(h/h_{\mathrm{a}}),$$
$$t^*_{\mathrm{x}} = (x^* - x_{\mathrm{a}})/w_{\mathrm{a}}, \quad t^*_{\mathrm{y}} = (y^* - y_{\mathrm{a}})/h_{\mathrm{a}},$$
$$t^*_{\mathrm{w}} = \log(w^*/w_{\mathrm{a}}), \quad t^*_{\mathrm{h}} = \log(h^*/h_{\mathrm{a}}),$$

where x, y, w, and h denote the box's center coordinates and its width and height.

where x, y, w, and h denote the box's center coordinates and its width and height.

- Variables x, x_a and $x^{\times}$ are for the predicted box, anchor box, and ground truth box respectively (likewise for y,w and h).
- This can be thought of as bounding-box regression from an anchor box to a nearby ground-truth box.
- Fast R-CNN achieves bounding-box regression by a different manner from previous RoI based (Region of Interest) methods.
- Unlike earlier methods where bounding-box regression is performed on features pooled from arbitrarily sized RoIs, and the regression weights are shared by all region sizes,
- In Fast R-CNN, the features used for regression are of the same spatial size (3 3) on the feature maps. To account for varying sizes, a set of k bounding-box regressors are learned. Each regressor is responsible for one scale and one aspect ratio, and the k regressors do not share weights.

Sharing features for RPN and Fast R-CNN

- The RPN is trained end-to-end by back propagation and stochastic gradient descent (SGD). The “image-centric” sampling strategy from Fast R-CNN is adopted to train this network.
- Each mini-batch arises from a single image that contains many positive and negative example anchors.
- The detection network is also adopted from Fast R-CNN.
- Both RPN and Fast R-CNN, trained independently, will modify their convolutional layers in different ways. We therefore need to develop a technique that allows for sharing convolutional layers between the two networks, rather than learning two separate net works. The authors of Faster R-CNN discuss three ways for training networks with shared features:
  - Alternate training
  - Approximate joint training
  - Non-approximate joint training (not in the scope of this paper)

- Alternate training
  - In this solution, we first train RPN, and use the proposals to train Fast R-CNN.
  - The network tuned by Fast R-CNN is then used to initialize RPN, and this process is iterated.
  - This is the solution that is used in all experiments in the Faster R-CNN paper.
- Approximate joint training
  - In this solution, the RPN and Fast R-CNN networks are merged into one network during training.
  - In each SGD iteration, the forward pass generates region proposals which are treated just like fixed, pre-computed proposals when training a Fast R-CNN detector.
  - The backward propagation takes place as usual, where for the shared layers the backward propagated signals from both the RPN loss and the Fast R-CNN loss are combined.
  - This solution ignores the derivative w.r.t. the proposal boxes' coordinates that are also network responses, so is approximate.
  - This produces close results yet reduces the training time by about 25-50% compared with alternating training.

4-Step Alternate Training

- The authors adopt a pragmatic 4-step training algorithm to learn shared features via alternating optimization.
  a. In the first step, we train the RPN as described earlier under “Training RPNs”. This network is initialized with an ImageNet-pre-trained model and fine-tuned end-to-end for the region proposal task.
  b. In the second step, we train a separate detection network by Fast R-CNN using the proposals generated by the step-1 RPN. This detection network is also initialized by the ImageNet-pre-trained model. At this point the two networks do not share convolutional layers.
  c. In the third step, we use the detector network to initialize RPN training, but we fix the shared convolutional layers and only fine-tune the layers unique to RPN. Now the two networks share convolutional layers.
  d. Finally, keeping the shared convolutional layers fixed, we fine-tune the unique layers of Fast R-CNN. As such, both networks share the same convolutional layers and form a unified network.

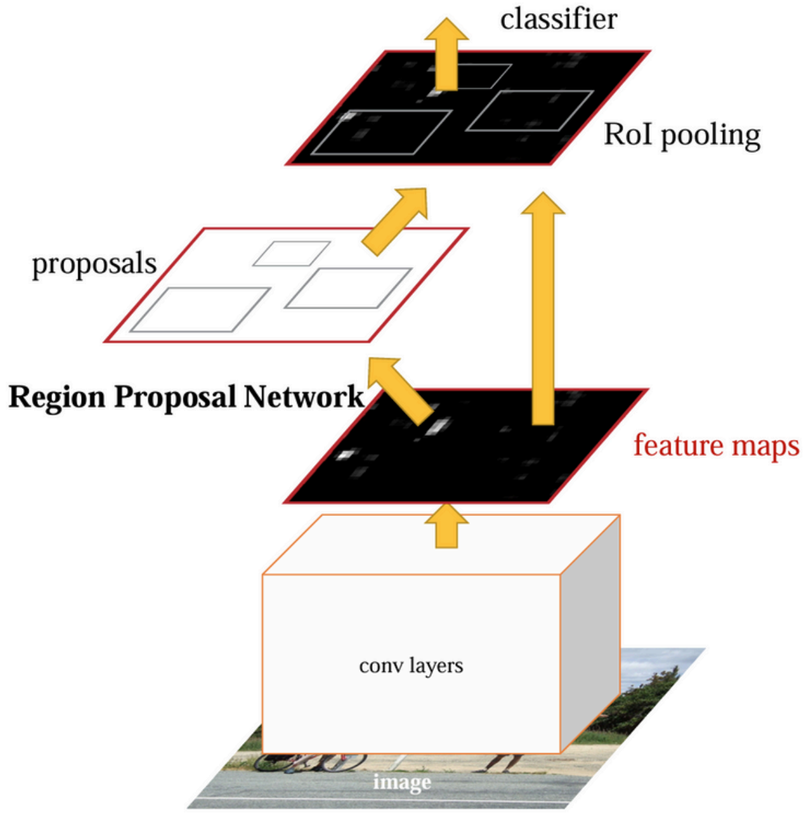


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

Implementation details

- We train and test both region proposal and object detection networks on images of a single scale. Images are re-scaled such that their shorter side is s = 600 pixels. Multi-scale feature extraction (using an image pyramid) may improve accuracy but does not exhibit a good speed-accuracy trade-off as in Fast R-CNN.
- For anchors, we use 3 scales with box areas of 128^2, 256^2, and 512^2 pixels, and 3 aspect ratios of 1:1, 1:2, and 2:1.
- The anchor boxes that cross image boundaries need to be handled with care. During training, we ignore all cross-boundary anchors so they do not contribute to the loss. During testing, however, we still apply the fully convolutional RPN to the entire image. This may generate cross-boundary proposal boxes, which we clip to the image boundary.
- Some RPN proposals highly overlap with each other.To reduce redundancy, we adopt non-maximum suppression (NMS) on the proposal regions based on their cls scores. We fix the IoU threshold for NMS at 0.7, which leaves us about 2000 proposal regions per image.

Table 2: Detection results on **PASCAL VOC 2007 test set** (trained on VOC 2007 trainval). The detectors are Fast R-CNN with ZF, but using various proposal methods for training and testing.

train-time region proposals		**test-time region proposals**		
method	# boxes	method	# proposals	mAP (%)
SS	2000	SS	2000	58.7
EB	2000	EB	2000	58.6
RPN+ZF, shared	2000	RPN+ZF, shared	300	**59.9**
ablation experiments follow below				
RPN+ZF, unshared	2000	RPN+ZF, unshared	300	58.7
SS	2000	RPN+ZF	100	55.1
SS	2000	RPN+ZF	300	56.8
SS	2000	RPN+ZF	1000	56.3
SS	2000	RPN+ZF (no NMS)	6000	55.2
SS	2000	RPN+ZF (no *cls*)	100	44.6
SS	2000	RPN+ZF (no *cls*)	300	51.4
SS	2000	RPN+ZF (no *cls*)	1000	55.8
SS	2000	RPN+ZF (no *reg*)	300	52.1
SS	2000	RPN+ZF (no *reg*)	1000	51.3
SS	2000	RPN+VGG	300	59.2

Table 5: **Timing** (ms) on a K40 GPU, except SS proposal is evaluated in a CPU. "Region-wise" includes NMS, pooling, fully-connected, and softmax layers. See our released code for the profiling of running time.

model	system	conv	proposal	region-wise	total	rate
VGG	SS + Fast R-CNN	146	1510	174	1830	0.5 fps
VGG	RPN + Fast R-CNN	141	**10**	47	**198**	**5 fps**
ZF	RPN + Fast R-CNN	31	**3**	25	**59**	**17 fps**

Table 3: Detection results on **PASCAL VOC 2007 test set**. The detector is Fast R-CNN and VGG-16. Training data: "07": VOC 2007 trainval, "07+12": union set of VOC 2007 trainval and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2000. †: this number was reported in [2]; using the repository provided by this paper, this result is higher (68.1).

method	# proposals	data	mAP (%)
SS	2000	07	66.9†
SS	2000	07+12	70.0
RPN+VGG, unshared	300	07	68.5
RPN+VGG, shared	300	07	69.9
RPN+VGG, shared	300	07+12	**73.2**
RPN+VGG, shared	300	COCO+07+12	**78.8**

Table 4: Detection results on **PASCAL VOC 2012 test set**. The detector is Fast R-CNN and VGG-16. Training data: "07": VOC 2007 trainval, "07++12": union set of VOC 2007 trainval+test and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2000. †: http://host.robots.ox.ac.uk:8080/anonymous/HZJTQA.html. ‡: http://host.robots.ox.ac.uk:8080/anonymous/YNPLXB.html. §: http://host.robots.ox.ac.uk:8080/anonymous/XEDH10.html.

method	# proposals	data	mAP (%)
SS	2000	12	65.7
SS	2000	07++12	68.4
RPN+VGG, shared†	300	12	67.0
RPN+VGG, shared‡	300	07++12	**70.4**
RPN+VGG, shared§	300	COCO+07++12	**75.9**

Table 8: Detection results of Faster R-CNN on PASCAL VOC 2007 test set using **different settings of anchors**. The network is VGG-16. The training data is VOC 2007 trainval. The default setting of using 3 scales and 3 aspect ratios (69.9%) is the same as that in Table 3.

settings	anchor scales	aspect ratios	mAP (%)
1 scale, 1 ratio	128^2	1:1	65.8
	256^2	1:1	66.7
1 scale, 3 ratios	128^2	{2:1, 1:1, 1:2}	68.8
	256^2	{2:1, 1:1, 1:2}	67.9
3 scales, 1 ratio	$\{128^2, 256^2, 512^2\}$	1:1	**69.8**
3 scales, 3 ratios	$\{128^2, 256^2, 512^2\}$	{2:1, 1:1, 1:2}	**69.9**

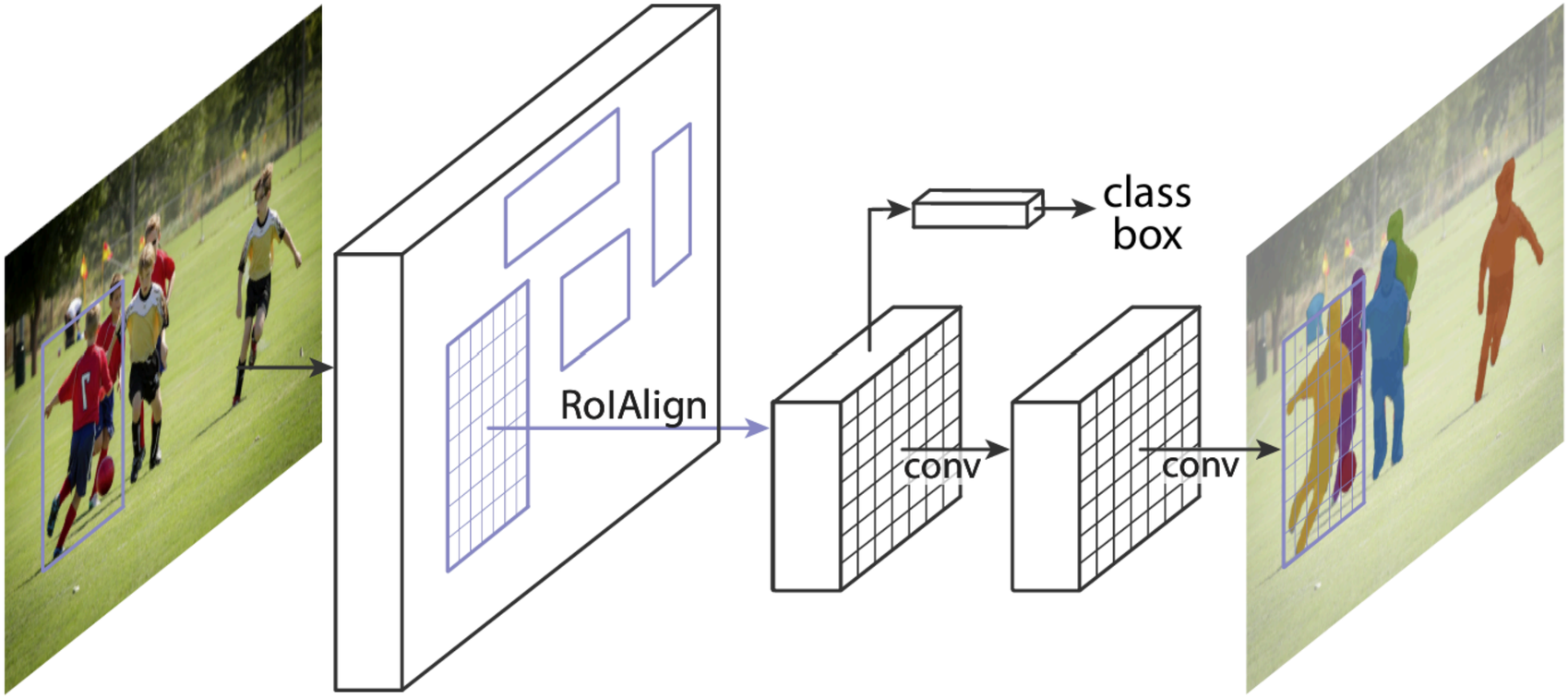
Table 9: Detection results of Faster R-CNN on PASCAL VOC 2007 test set using **different values of** λ in Equation (1). The network is VGG-16. The training data is VOC 2007 trainval. The default setting of using $\lambda = 10$ (69.9%) is the same as that in Table 3.

λ	0.1	1	10	100
mAP (%)	67.2	68.9	69.9	69.1

Intro to Mask R-CNN

- Instance segmentation is challenging because it requires the correct detection of all objects in an image while also precisely segmenting each instance.
- It therefore combines elements from the classical computer vision tasks of object detection, where the goal is to classify individual objects and localize each using a bounding box, and semantic segmentation, where the goal is to classify each pixel into a fixed set of categories without differentiating object instances.
- Mask R-CNN extends Faster R-CNN by adding a branch for predicting segmentation masks on each Region of Interest (RoI), in parallel with the existing branch for classification and bounding box regression.
- The mask branch is a small FCN applied to each RoI, predicting a segmentation mask in a pixel-to-pixel manner.
- Mask R-CNN is simple to implement and train given the Faster R-CNN framework, which facilitates a wide range of flexible architecture designs.
- Additionally, the mask branch only adds a small computational overhead, enabling a fast system and rapid experimentation.

- Faster R CNN was not designed for pixel-to-pixel alignment between network inputs and outputs. This is most evident in how RoIPool, the de facto core operation for attending to instances, performs coarse spatial quantization for feature extraction.
- To fix the misalignment, the authors of Mask R-CNN propose a simple, quantization-free layer, called RoIAlign, that preserves exact spatial locations.
- Despite being a seemingly minor change, RoIAlign has a large impact: it improves mask accuracy by relative 10% to 50%, showing bigger gains under stricter localization metrics.
- Second, the authors found it essential to decouple mask and class prediction: we predict a binary mask for each class independently, without competition among classes, and rely on the network's RoI classification branch to predict the category.
- In contrast, FCNs usually perform per-pixel multi-class categorization, which couples segmentation and classification, leading to competition among classes and based on the experiments works poorly for instance segmentation.


RoIAlign
class
box
conv
conv

Mask R-CNN Architecture

- Mask R-CNN adopts the same two-stage procedure, with an identical first stage (which is RPN). In the second stage, in parallel to predicting the class and box offset, Mask R-CNN also outputs a binary mask for each RoI.
- Formally, during training, we define a multi-task loss on each sampled RoI as $L = L_{cls} + L_{box} + L_{mask}$
- The classsification loss L_{cls} and bounding-box loss L_{box} are identical as those defined in Fast R-CNN.
- The mask branch has a Km^2 dimensional output for each RoI, which encodes K binary masks of resolution m x m, one for each of the K classes. To this we apply a per-pixel sigmoid, and define L_{mask} as the average binary cross-entropy loss. For an RoI associated with ground-truth class k, L_{mask} is only defined on the k-th mask (other mask outputs do not contribute to the loss).
- This definition of Lmask allows the network to generate masks for every class without competition among classes. Mask R-CNN relies on the dedicated classification branch to predict the class label used to select the output mask. **This decouples mask and class prediction.**

Mask Representation

- A mask encodes an input object’s spatial layout. Thus, unlike class labels or box offsets that are inevitably collapsed into short output vectors by fully-connected (fc) layers, extracting the spatial structure of masks can be addressed naturally by the pixel-to-pixel correspondence provided by convolutions.
- Specifically, we predict an m x m mask from each RoI using an FCN. This allows each layer in the mask branch to maintain the explicit m m object spatial layout without collapsing it into a vector representation that lacks spatial dimensions.
- This pixel-to-pixel behavior requires our RoI features, which themselves are small feature maps, to be well aligned to faithfully preserve the explicit per-pixel spatial correspondence.
- This requirement motivates the authors to develop an RoI Align layer that plays a key role in mask prediction as explained next.

RoI Align

- RoIPool is a standard operation for extracting a small feature map (e.g., 7 7) from each RoI. RoIPool first quantizes a floating-number RoI to the discrete granularity of the feature map, this quantized RoI is then subdivided into spatial bins which are themselves quantized, and finally feature values covered by each bin are aggregated (usually by max pooling).
- Quantization is performed, e.g., on a continuous coordinate x by computing [x/16], where 16 is a feature map stride and [.] is rounding; likewise, quantization is performed when dividing into bins (e.g., 7 x 7).
- These quantizations introduce misalignments between the RoI and the extracted features and has a large negative effect on predicting pixel-accurate masks.
- RoI Align avoids any quantization of the RoI boundaries or bins (i.e., we use x/16 instead of [x/16]). Align uses bi-linear interpolation to compute the exact values of the input features at four regularly sampled locations in each RoI bin, and aggregate the result (using max or average).

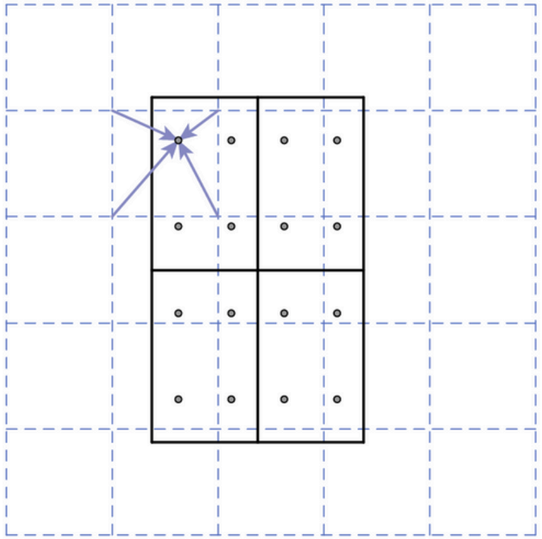


Figure 3. **RoIAlign:** The dashed grid represents a feature map, the solid lines an RoI (with 2×2 bins in this example), and the dots the 4 sampling points in each bin. RoIAlign computes the value of each sampling point by bilinear interpolation from the nearby grid points on the feature map. No quantization is performed on any coordinates involved in the RoI, its bins, or the sampling points.

- Results are not sensitive to the exact sampling locations, or how many points are sampled, as long as no quantization is performed.

Implementation Details

Training

- As in Fast R-CNN, an RoI is considered positive if it has IoU with a ground-truth box of at least 0.5 and negative otherwise. The mask loss Lmask is defined only on positive RoIs. The mask target is the intersection between an RoI and its associated ground-truth mask.
- Mask R-CNN adopts image-centric training from Fast R-CNN. Images are resized such that their scale (shorter edge) is 800 pixels. Each mini-batch has 2 images per GPU and each image has N sampled RoIs, with a ratio of 1:3 of positive to negatives as in Fast R-CNN. N is 64 for the C4 backbone and 512 for FPN.
- We train on 8 GPUs (so effective mini batch size is 16) for 160k iterations, with a learning rate of 0.02 which is decreased by 10 at the 120k iteration. A weight decay of 0.0001 and momentum of 0.9 is used. With ResNeXt, we train with 1 image per GPU and the same number of iterations, with a starting learning rate of 0.01.
- The RPN anchors span 5 scales and 3 aspect ratios, following the FPN paper. For convenient ablation, RPN is trained separately and does not share features with Mask R-CNN, unless specified. For every entry in the paper, RPN and Mask R-CNN have the same backbones and so they are shareable.

Inference

- At test time, the proposal number is 300 for the C4 backbone and 1000 for FPN.
- We run the box prediction branch on these proposals, followed by non-maximum suppression.
- The mask branch is then applied to the highest scoring 100 detection boxes.
- Although this differs from the parallel computation used in training, it speeds up inference and improves accuracy (due to the use of fewer, more accurate RoIs). The mask branch can predict K masks per RoI, but we only use the k-th mask, where k is the predicted class by the classification branch. The m x m floating-number mask output is then resized to the RoI size, and binarize data threshold of 0.5.

net-depth-features	AP	AP_{50}	AP_{75}
ResNet-50-C4	30.3	51.2	31.5
ResNet-101-C4	32.7	54.2	34.3
ResNet-50-FPN	33.6	55.2	35.3
ResNet-101-FPN	35.4	57.3	37.5
ResNeXt-101-FPN	**36.7**	**59.5**	**38.9**

(a) **Backbone Architecture**: Better backbones bring expected gains: deeper networks do better, FPN outperforms C4 features, and ResNeXt improves on ResNet.

	AP	AP_{50}	AP_{75}
softmax	24.8	44.1	25.1
sigmoid	**30.3**	**51.2**	**31.5**
	+5.5	*+7.1*	*+6.4*

(b) **Multinomial *vs*. Independent Masks** (ResNet-50-C4): *Decoupling* via per-class binary masks (sigmoid) gives large gains over multinomial masks (softmax).

	align?	bilinear?	agg.	AP	AP_{50}	AP_{75}
RoIPool [12]			max	26.9	48.8	26.4
RoIWarp [10]		✓	max	27.2	49.2	27.1
		✓	ave	27.1	48.9	27.1
RoIAlign	✓	✓	max	**30.2**	**51.0**	**31.8**
	✓	✓	ave	**30.3**	**51.2**	**31.5**

(c) **RoIAlign** (ResNet-50-C4): Mask results with various RoI layers. Our RoIAlign layer improves AP by ~3 points and AP_{75} by ~5 points. Using proper alignment is the only factor that contributes to the large gap between RoI layers.

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MNC [10]	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [26] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [26] +OHEM	ResNet-101-C5-dilated	33.6	54.5	-	-	-	-
Mask R-CNN	ResNet-101-C4	33.1	54.9	34.8	12.1	35.6	51.1
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	ResNeXt-101-FPN	**37.1**	**60.0**	**39.4**	**16.9**	**39.9**	**53.5**

Table 1. **Instance segmentation** *mask* AP on COCO `test-dev`. MNC [10] and FCIS [26] are the winners of the COCO 2015 and 2016 segmentation challenges, respectively. Without bells and whistles, Mask R-CNN outperforms the more complex FCIS+++, which includes multi-scale train/test, horizontal flip test, and OHEM [38]. All entries are *single-model* results.

	backbone	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{bb}_S	AP^{bb}_M	AP^{bb}_L
Faster R-CNN+++ [19]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [27]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [21]	Inception-ResNet-v2 [41]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [39]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
Faster R-CNN, RoIAlign	ResNet-101-FPN	37.3	59.6	40.3	19.8	40.2	48.8
Mask R-CNN	ResNet-101-FPN	38.2	60.3	41.7	20.1	41.1	50.2
Mask R-CNN	ResNeXt-101-FPN	**39.8**	**62.3**	**43.4**	**22.1**	**43.2**	51.2

Table 3. **Object detection** *single-model* results (bounding box AP), *vs.* state-of-the-art on `test-dev`. Mask R-CNN using ResNet-101-FPN outperforms the base variants of all previous state-of-the-art models (the mask output is ignored in these experiments). The gains of Mask R-CNN over [27] come from using RoIAlign (+1.1 AP^{bb}), multitask training (+0.9 AP^{bb}), and ResNeXt-101 (+1.6 AP^{bb}).

Figure 5. More results of **Mask R-CNN** on COCO test images, using ResNet-101-FPN and running at 5 fps, with 35.7 mask AP (Table 1).

Thank You